Adriano João Vieira Chora

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Arquitetura para Sistema Distribuído de Manufatura Baseado em Agentes Utilizando Arduinos

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Doutor Associado com Agregação, UNINOVA/CTS, FCT/UNL

Co-orientador: André Dionísio Bettencourt da Silva Rocha, Professor Doutor Auxiliar Convidado, UNINOVA/CTS, FCT/UNL

Júri

Presidente: Prof.ª Doutora Anikó Costa
Arguentes: Prof. Doutor Ricardo Peres
Vogais: Prof. Doutor André Rocha

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2020

Arquitetura para Sistema Distribuído de Manufatura Baseado em Agentes Utilizando Arduinos

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

À minha família

Agradecimentos

Nesta secção gostaria de agradecer a todos os que tornaram a realização desta dissertação possível.

Em primeiro gostaria de agradecer ao meu orientador, Professor Doutor José Barata, pela oportunidade que me deu de realizar esta dissertação numa área que me suscita bastante interesse.

Ao meu co-orientador, Professor Doutor André Rocha, gostaria de deixar um sentido agradecimento por toda a paciência, apoio e disponibilidade na realização deste trabalho, sem os quais não teria sido possível a realização do mesmo.

Gostaria de agradecer também aos meus amigos de faculdade, não só pelo apoio na realização desta dissertação, mas também por todos os momentos vividos ao longo do curso.

Um agradecimento à Tania, pela paciência, apoio e carinho demonstrados todos os dias durante este capítulo e que muito me ajudaram a concluir esta etapa.

Por fim gostaria de agradecer aos meus pais, ao meu irmão e avós, que me educaram e que sempre me deram o seu apoio incondicional. Um especial agradecimento à minha mãe por todas as conversas, conselhos e pelo esforço enorme que sempre fez para que eu pudesse chegar até aqui.

Resumo

Nos últimos tempos tem havido uma necessidade das empresas de manufatura que adotam um modelo de produção em massa, de evoluírem para um novo modelo capaz de responder às exigências atuais dos mercados, caracterizadas pela elevada personalização de produtos.

Alguns sistemas têm aparecido ao longo dos anos numa tentativa de responder a esta necessidade. No entanto a sua implementação tem sido difícil pois o *hardware* necessário é de custo muito elevado ou inexistente. Estes sistemas estão incluídos na quarta revolução industrial, também conhecida por Indústria 4.0. Este conceito, assente na Internet das Coisas e nos Sistemas Cíber-Físicos, procura reunir os princípios que devem ser aplicados na criação de um sistema flexível capaz de implementar este novo modelo.

Assim, esta dissertação propõe uma arquitetura em que o controlo é dividido entre dispositivos baseados na Internet das Coisas, responsáveis pelos processos em tempo real, e um sistema multiagente, o qual gere todos os outros processos.

É proposta a utilização de dispositivos de baixo custo baseados em Arduino como facilitadores da implementação de sistemas capazes de lidar com a adição e remoção de componentes sem necessidade de reprogramação, também conhecido como *Plug&Produce*, ao mesmo tempo que se utiliza um sistema multiagente, na tentativa de obter um sistema distribuído, integrável, reconfigurável e descentralizado.

No final é apresentado um conjunto de testes, realizados com dois tipos de microcontroladores, em que estes são adicionados e removidos do sistema de forma rápida sem que este tenha de ser reprogramado.

Palavras-chave: Indústria 4.0; Internet das Coisas; Sistemas Cíber-Físicos; Arduino; *Plug&Produce*; Sistema Multiagente; Sistema Distribuído de Controlo.

Abstract

In recent years there has been a need for manufacturing companys that adopt a mass production model, to evolve to a new model capable of responding to the current market demands characterized by high product customization.

Over the years there have appeared some systems that aimed to respond to this need. However their implementation has been difficult due to the lack of available hardware or his high cost. This systems are included in the fourth industrial revolution, also known as Industry 4.0. This concept, which was built upon the concepts of Internet of Things and Cyber-Physical Systems, searches for the principles that must be applied to the creation of a flexible system capable of implementing this new model.

Thus, this thesis proposes an architecture where control is divided between devices based in the Internet of Things, which are responsable for the real-time processes, and a multi-agent system that manages all of the other processes.

The utilization of low-cost devices based in Arduino is proposed as facilitators of an implementation of systems that are capable of managing the adding and removal of devices whitout a need for reprogramming, also known as Plug&Produce, while at the same time using multi-agent systems in an attempt to obtain a distributed, integrable, reconfigurable and descentralized system.

In the end, a set of trials, conducted resorting to two types of microcontrollers is presented, where this microcontrollers are added and removed from the system in a quick process, without the need of reprogramming it.

Keywords: Industry 4.0; Internet of Things; Cyber-Physical Systems; Arduino; Plug&Produce; Multi-Agent System; Distributed Control System.

Índice

Lista de Figuras **xv**

Lista de Tabelas **xvii**

Siglas **xix**

1 Introdução **1**
1.1 Perguntas de Investigação e Hipóteses 2
1.2 Visão Geral do Trabalho Realizado 2
1.3 Principais Contribuições 2

2 Estado da Arte **5**
2.1 Indústria 4.0 5
2.2 Sistemas Cíber-Físicos 7
2.3 Novos Paradigmas de Produção 8
2.3.1 Sistemas Flexíveis de Manufatura 8
2.3.2 Sistemas Reconfiguráveis de Manufatura 9
2.3.3 Paradigmas Baseados em Sistemas Multiagente 10
2.4 Conclusões Gerais 14

3 Conceitos de Suporte **15**
3.1 *Web Services* e Arquiteturas Orientadas a Serviços 15
3.2 Internet of Things 16
3.3 Computação e Manufatura em Nuvem 18
3.4 JADE 19
3.5 FIPA 20
3.5.1 FIPA Request 21
3.5.2 FIPA ContractNet 22
3.6 Protocolos de Transporte 23
3.6.1 UDP 24
3.6.2 TCP 24
3.7 Sockets 24

4 Arquitetura **27**

4.1 Arquitetura do Sistema 27
4.2 Arquitetura do Sistema Multiagente 31
4.2.1 *Product Agent* (PA) 33
4.2.2 *Deployment Agent* (DA) 35
4.2.3 *Resource Agent* (RA) 35
4.3 Dispositivo IoT 36
4.3.1 Integração de um dispositivo 37
4.3.2 Execução de uma habilidade 38
4.3.3 Remoção de um dispositivo 39

5 Implementação **41**
5.1 Sistema Multiagente 42
5.1.1 *Directory Facilitator* 42
5.1.2 Classes Auxiliares 42
5.1.3 *Product Agent* 43
5.1.4 *Resource Agent* 44
5.1.5 *Deployment Agent* 45
5.1.6 Comunicação entre Agentes 45
5.2 Dispositivo IoT 47
5.3 Comunicação entre sistema multiagente e dispositivo 48
5.3.1 Ligação do dispositivo ao sistema multiagente 48
5.3.2 Execução de uma habilidade 49
5.3.3 Remoção de um dispositivo do sistema multiagente 50

6 Validação e Testes **53**
6.1 Configuração do Ambiente de Teste 53
6.2 Teste de Velocidade de Conexão 56
6.3 Testes de Desempenho do Sistema 57
6.3.1 Teste com Configuração de Linha 1 57
6.3.2 Teste com Configuração de Linha 2 62

7 Conclusões e Trabalho Futuro **67**
7.1 Conclusões 67
7.2 Trabalho Futuro 68

Bibliografia **69**

Lista de Figuras

3.1 Protocolo FIPA Request [40]. 22
3.2 Protocolo FIPA ContractNet [41]. 23

4.1 Arquitetura simples do sistema. 28
4.2 Estado 1. 29
4.3 Estado 2. 29
4.4 Estado 3. 30
4.5 Estado 4. 30
4.6 Arquitetura completa do sistema. 31
4.7 Arquitetura do SMA. 32
4.8 Fluxograma do PA. 34
4.9 Fluxograma do DA. 35
4.10 Fluxograma do RA. 36
4.11 Exemplo de um módulo de produção com dois recursos físicos. 37
4.12 Fluxograma da integração de um dispositivo. 38
4.13 Fluxograma da execução de uma habilidade no dispositivo. 38
4.14 Fluxograma do controlo feito ao dispositivo. 39

5.1 Visão geral da implementação. 41
5.2 Classe *Constants* 42
5.3 Classe *DFInteraction* 42
5.4 Classe *ProductAgent* 43
5.5 Classe *ResourceAgent* 44
5.6 Classe *DeploymentAgent* 45
5.7 Diagrama de sequência da negociação entre PA e RA. 46
5.8 Diagrama de sequência do pedido de transporte do PA ao TA. 46
5.9 Diagrama de sequência do pedido de execução de uma habilidade do PA ao RA. 47
5.10 Diagrama de sequência do pedido de ligação de um dispositivo ao DA. 49
5.11 Diagrama de sequência do pedido de execução de uma habilidade do RA a um dispositivo. 50
5.12 Diagrama de sequência do controlo do RA a um dispositivo. 51

6.1 Conjunto Arduino Uno e Arduino Ethernet Shield 2. 54

6.2 ETHERNET PLC M-DUINO 58+ e distribuição de zonas. 55
6.3 Kit de Demonstração da Staudinger GMBH. 55
6.4 Passadeira e Módulos de Produção 56
6.5 Configuração de linha 1. 58
6.6 Execução da H_A no kit. 58
6.7 Fluxograma do plano de execução de H_A no kit de demonstração. 59
6.8 Evolução do primeiro conjunto de 5 produtos no simulador. 60
6.9 Evolução do segundo conjunto de 5 produtos no simulador. 60
6.10 Evolução do terceiro conjunto de 5 produtos no simulador. 61
6.11 Configuração de linha 2. 62
6.12 Fluxograma do plano de execução de H_C no kit de demonstração. 63
6.13 Evolução do primeiro produto no simulador. 64
6.14 Evolução do segundo produto no simulador. 65

Lista de Tabelas

4.1 Descrição dos Agentes. 33

6.1 Especificações do Arduino Uno [44] 53
6.2 Especificações do M-DUINO PLC ARDUINO ETHERNET 58 I/O's ANALOG/-DIGITAL PLUS [45]. 54
6.3 Resultados do teste de velocidade de conexão. 57
6.4 Resultados do primeiro conjunto de 5 produtos. 61
6.5 Resultados do segundo conjunto de 5 produtos. 61
6.6 Resultados do terceiro conjunto de 5 produtos. 61

Siglas

ADACOR *ADAptive holonic COntrol aRchitecture*.

AOS Arquitetura Orientada a Serviços.

DA *Deployment Agent*.

DF *Directory Facilitator*.

FIPA *Foundation for Intelligent Physical Agents*.

HTTP *HyperText Transfer Protocol*.

IDEAS *Industrial Agents for the Fast Deployment of Evolvable Assembly Systems*.

IIoT *Industrial Internet of Things*.

IoT *Internet of Things*.

IP *Internet Protocol*.

JADE *Java Agent Development Framework*.

M2M *Machine-to-Machine*.

MASCADA *Manufacturing Control Systems Capable of Managing Production Change and Disturbances*.

NIST *National Institute of Standards and Technology*.

PA *Product Agent*.

PROSA *Product Resource Order Staff Architecture*.

RA *Resource Agent*.

REST *Representational State Transfer*.

SBM Sistema Biónico de Manufatura.

SCF Sistemas Cíber-Físicos.

SCFP Sistemas Cíber-Físicos de Produção.

SDM Sistema Distribuído de Manufatura.

SEP Sistema Evolutivo de Produção.

SFM Sistema Flexível de Manufatura.

SHM Sistema Holónico de Manufatura.

SIRENA *Service Infrastructure for Real-time Embedded Networked Applications*.

SMA Sistema Multiagente.

SOAP *Simple Project Access Protocol*.

SRM Sistema Reconfigurável de Manufatura.

TA *Transport Agent*.

TCP *Transmission Control Protocol*.

TIC Tecnologias de Informação e Comunicação.

UDP *User Datagram Protocol*.

WSDL *Web Services Description Language*.

XML *Extensible Markup Language*.

Capítulo 1

Introdução

Desde a popularização do conceito de produção em massa por Henry Ford no início do séc. XX que as empresas de manufatura têm baseado a sua produção num modelo *make-to-stock*, conseguindo elevadas taxas de produção de produtos padrão a um custo baixo. No entanto, nas últimas décadas, com as constantes inovações tecnológicas e com a globalização, este modelo começou a ficar desatualizado e a procura por novos modelos de manufatura tem sido constante.

Os novos paradigmas de produção têm de ser capazes de responder às exigências dos consumidores, caracterizadas pela elevada personalização dos produtos e por alterações constantes às quantidades e tipos de produtos a serem produzidos. Estes novos sistemas têm de ser flexíveis e capazes de se adaptar rapidamente às constantes mudanças nas exigências de mercado. Tem de haver uma mudança do tradicional *make-to-stock* para um modelo versátil *make-to-order*.

Assim, desde há algum tempo, têm surgido vários novos paradigmas de produção, muitos deles baseados em sistemas distribuídos, que visam suportar este tipo de características. No entanto a sua implementação apresenta alguns desafios, pois o *hardware* necessário de modo a suportar a parte de controlo é inexistente ou de custo muito elevado.

A presente dissertação apresenta uma contribuição de forma a tentar a resolução deste problema. Ao retirar o controlo de tarefas complexas do *hardware*, torna possível aplicar as funcionalidades dos sistemas de controlo distribuídos sem as restrições ao nível do *hardware*.

1.1 Perguntas de Investigação e Hipóteses

Tendo como base o problema apresentado anteriormente e as necessidades atuais da indústria, assim como a solução proposta em cima, é possível formular uma questão:

- Que arquitetura poderá ser desenhada de forma a integrar módulos de produção simples de baixo custo, não envolvidos no processo de decisão, de forma rápida e eficiente, num sistema de controlo distribuído e adaptável?

Assim, é proposto como hipótese, um sistema baseado numa arquitetura multiagente, caracterizada pela sua autonomia e adaptabilidade. São ainda utilizados dispositivos de controlo de baixo custo, os quais ligados fisicamente a recursos, formam módulos de produção. Assim, o sistema multiagente (SMA) tem a capacidade de comunicar com um ou mais módulos, possibilitando a sua adição e remoção, sem que com isso seja necessário parar as linhas ou haja a necessidade de reprogramação.

1.2 Visão Geral do Trabalho Realizado

Depois de consideradas as questões e hipóteses apresentadas anteriormente, foi proposta uma arquitetura baseada num SMA, fazendo uso das características associadas a este tipo de sistemas. Este sistema, executado numa máquina com capacidade para processar as tarefas de controlo complexas de forma ágil, abstrai produtos, recursos e sistema de transporte. Estas entidades trabalham em conjunto, de modo a obter um processo de produção eficaz.

Os módulos de produção são compostos por um dispositivo de controlo baseado em Arduino e um ou mais recursos físicos. Estes são ligados através de uma rede *Ethernet* com a máquina que executa o SMA. Assim, o dispositivo de controlo tem de ser capaz de se ligar a um ou mais recursos, através de entradas e saídas, suportar comunicação *Ethernet* e operar a uma tensão de 24V, sendo esta a tensão de operação da linha de montagem.

O sistema foi posteriormente testado num ambiente virtual e depois com recurso a um kit de demonstração, de modo a validar a hipótese apresentada.

1.3 Principais Contribuições

A solução apresentada nesta dissertação pretende explorar a utilização de *hardware* de custo mais acessível na implementação de um sistema de produção reconfigurável, eficiente e preparado a responder às exigências atuais dos consumidores. Isto poderá resultar numa adoção de mais soluções deste género, ficando disponível para mais empresas.

A facilidade de integração e remoção de módulos de produção no sistema faz com que rapidamente a linha de produção seja reconfigurada sem que seja necessário interromper

o processo de produção. Isto aumenta a capacidade de personalização de produtos, sem reduzir drasticamente a produtividade da linha.

De realçar que a implementação escolhida para a arquitetura aqui proposta é apenas uma de muitas possíveis, ficando assim as linhas principais a seguir para a obtenção de um sistema deste tipo, deixando ao mesmo tempo liberdade na escolha do *hardware*.

Capítulo 2

Estado da Arte

Tal como referido, a Indústria 4.0, assente nos sistemas cíber-físicos, procura reunir os princípios necessários à implementação de um sistema capaz de responder às novas exigências dos mercados. No entanto, alguns destes princípios não são novos, tendo já sido explorados por alguns sistemas, os quais foram evoluindo ao longo dos anos, sendo hoje em dia, ainda que com algumas limitações, utilizados para implementar sistemas distribuídos e adaptáveis. Estes conceitos são a base da qual esta dissertação parte na tentativa de oferecer uma contribuição para ultrapassar estas limitações.

2.1 Indústria 4.0

A Indústria 4.0 é uma iniciativa introduzida pelo governo alemão. O objetivo desta iniciativa é a transformação da manufatura industrial através da digitalização e da exploração dos potenciais das novas tecnologias. As tecnologias chave para a Indústria 4.0 são a computação em nuvem, a *Big Data* e a *Internet of Things* (IoT). Um sistema de produção deste tipo é flexível e permite a individualização e personalização de produtos.

A primeira revolução industrial começou no final do século 18 e foi caracterizada pela passagem do trabalho manual para os primeiros processos de manufatura com a utilização de máquinas a vapor. A segunda revolução industrial começou no ínicio do século 20 e foi caracterizada pela industrialização e a produção em massa, apenas permitidas pelo aparecimento da energia elétrica. A terceira revolução começou na década de 70 e foi caracterizada pela digitalização e automação com recurso à microeletrónica. Atualmente estamos na quarta revolução industrial, a qual se deve ao desenvolvimento das Tecnologias de Informação e Comunicação (TIC).

A base da quarta revolução industrial é a automação inteligente de Sistemas Cíber-Físicos (SCF) com controlo descentralizado e conectividade avançada (IoT). O aparecimento destas novas tecnologias permite a passagem dos sistemas de automação clássicos com organização hierárquica para os sistemas de produção cíber-físicos auto-organizáveis. Esta passagem vai permitir o aparecimento da produção flexível de produtos personalizáveis em massa.

Hoje em dia, a produção industrial é caracterizada pela competição global e pela necessidade de adaptação rápida da produção aos requisitos de mercado em constante mudança. Estes requisitos podem ser cumpridos recorrendo à aplicação de coceitos genéricos como SCF e *Industrial Internet of Things (IIoT)* aos sistemas de produção industrial. Assim, a Indústria 4.0 é baseada na conexão dos blocos fundamentais dos SCF. Estes blocos são sistemas embutidos com controlo descentralizado e conectividade avançada, os quais recolhem e trocam informação em tempo real com o objetivo de monitorizar e otimizar os processos de produção.

No contexto da Indústria 4.0, os sistemas de manufatura são atualizados para um nível inteligente. A manufatura inteligente tira partido da informação e das tecnologias de manufatura para obter processos reconfiguráveis, flexíveis e inteligentes [1].

São fatores chave deste conceito a interoperabilidade e a conetividade. A interoperabilidade é uma das maiores vantagens da Indústria 4.0, representando a capacidade de dois sistemas trocarem informação e conhecimento. Um fluxo contínuo de informação deve ser estabelecido entre dispositivos e componentes, interação *Machine-to-Machine* (M2M), sistemas de manufatura e atores. Assim, produtos e fábricas conseguem conectar-se através da IIoT.

Existem ainda outras vantagens para a adoção deste conceito, incluindo: um *time-to-market* mais curto para os novos produtos, uma melhor resposta do cliente, a possibilidade da produção personalizável em massa sem aumentar significativamente os custos, uma maior flexibilidade no ambiente de trabalho e um uso mais eficiente de recursos naturais e energia .

O processo fundamental do sistema de produção da Indústria 4.0 é a conversão do digital para o físico num sistema reconfigurável de manufatura. Estes sistemas, os quais serão apresentados mais à frente, são capazes de adaptar os seus componentes de *hardware* e *software* às constantes mudanças dos requisitos de mercado.

A Indústria 4.0 traz mudanças disruptivas para as cadeias de abastecimento, modelos de negócio e processos de negócio. Os princípios deste conceito são a interoperabilidade, a virtualização, a descentralização e a moduloridade. Este consegue ainda providenciar mais flexibilidade, maior capacidade de personalização e custos reduzidos. A descentralização, a auto-otimização e a automação conseguem assistir os processos dinâmicos de

uma forma mais eficiente.

Os produtos, os processos, os controlos e a inteligência artificial baseados em multi-agente apresentam um processo de interoperabilidade. O fluxo de informação é coordenado e comunicado entre os participantes na manufatura e os agentes nos SCF. Assim, a tecnologia baseada em agentes, explicada mais à frente, é uma ferramenta apropriada para lidar com a complexidade e planeamento da manufatura na Indústria 4.0 [2][3].

2.2 Sistemas Cíber-Físicos

Os Sistemas Cíber-Físicos estão na origem de um desenvolvimento global e sem precedentes de produtos e serviços de vários domínios. Este conceito assenta numa nova geração de sistemas embutidos com capacidades de comunicação melhoradas e um elo inteligente e permanente entre o componente lógico e a sua parte física associada. São sistemas de entidades computacionais colaborativas, os quais estão conectados com o mundo físico envolvente e os seus processos, facultando e utilizando, ao mesmo tempo, serviços de processamento e de acesso a informação disponíveis na *Internet*. Os SCF podem ainda ser desenvolvidos para gerir *Big Data* e para tirar partido da interconetividade de máquinas, de modo a obter máquinas inteligentes, resilientes e auto-adaptáveis [4].

São ainda definidos como um conjunto de tecnologias utilizadas para gerir sistemas interconectáveis entre os seus recursos físicos e as suas capacidades computacionais. Com os recentes desenvolvimentos que resultaram numa maior disponibilidade e acessibilidade de sensores, sistemas de aquisição de dados e redes de computadores, a natureza competitiva da indústria atual faz com que a maioria das fábricas se vire para a implementação de metodologias de alta tecnologia.

O potencial dos SCF para mudar vários aspetos da vida é enorme. Conceitos como carros autónomos, edifícios inteligentes ou manufatura inteligente são apenas alguns exemplos. Para além disto, ao integrar SCF, na forma de Sistemas Cíber-Físicos de Produção (SCFP), com a produção, a logística e os serviços nas práticas industriais atuais, iria permitir transformar as fábricas atuais em fábricas da Indústria 4.0.

Os SCFP consistem em elementos e sub-sistemas autónomos e cooperativos, os quais estão conectados através de todos os níveis de produção, desde processos de máquinas até à produção e redes logísticas. Os SCFP possuem três características importantes:

- **Inteligência** — Os elementos têm a capacidade de adquirir informação do ambiente que os rodeia e atuar de forma autónoma;
- **Conetividade** — Têm a habilidade de configurar e utilizar conexões com os outros elementos do sistema de modo a cooperarem e colaborarem;
- **Responsivos** — Em relação a mudanças internas e externas.

Dentro das redes de abastecimento globais, a maquinaria, os sistemas de armazenamento e as instalações de produção serão incorporados na forma de SCFP. Estes irão trocar informação de forma autónoma, desencadeando ações e sendo controlados entre si de forma independente, de modo a atingir uma fábrica inteligente [5][6].

A utilização dos SCFP na Indústrtia 4.0 está diretamente ligada com o desenvolvimento das entidades computacionais, dos procedimentos relacionados com informação, da automação e tecnologia na manufatura e das TIC. Os SCFP têm a capacidade de dominar os sistemas de manufatura pela integração com os SCF, criando uma nova geração de indústria. Estes envolvem humanos, máquinas e produtos, ao mesmo tempo que combinam computação e processos físicos com o objetivo de tornar a produção mais eficiente em termos de custos e tempo [3].

2.3 Novos Paradigmas de Produção

Apesar do recente interesse nos conceitos dos SCF e da Indústria 4.0, é importante considerar que grande parte das ideias apresentadas por estes conceitos já estão presentes na comunidade científica há mais tempo. Assim, a este respeito, o trabalho nas áreas dos Sistemas Flexíveis de Manufatura, dos Sistemas Reconfiguráveis de Manufatura, dos Sistemas Holónicos de Manufatura, dos Sistemas Biónicos de Manufatura, dos Sistemas Evolutivos de Produção e dos Sistemas Multiagente pode ser considerado como o predecessor destes novos conceitos, sendo que utilizam os mesmos princípios.

2.3.1 Sistemas Flexíveis de Manufatura

Um sistema flexível de manufatura (SFM) consiste numa ou mais estações de processamento, interconectadas por um sistema automatizado de encaminhamento e armazenamento de materiais e que é controlado por um sistema de computador distribuído. É flexível na medida em que permite processar uma variedade de diferentes tipos de partes simultaneamente nas várias estações de trabalho e esta variedade de estilos de peças e quantidades de produção pode ser ajustada em resposta a mudanças na procura. Um sistema deste tipo é projetado para produzir produtos dentro de um intervalo definido de estilos, tamanhos e processos [7].

No entanto, existem algumas dificuldades na implementação deste tipo de sistemas. O carregamento de peças envolve decisões na atribuição de trabalho a diferentes conjuntos de máquina-ferramenta. Algumas abordagens a este tópico têm sido propostas ao longo dos anos.

- **Métodos de Programação Matemática e Analítica** — Relativamente a esta área foi criado e desenvolvido ao longo dos anos, por vários autores, um modelo MIP (*non-linear 0-1 mixed-integer programming*) que tem em conta vários fatores, tais como,

o encaminhamento de peças através das máquinas, a alocação de ferramentas, as limitações no número de ferramentas disponíveis, diferentes objetivos de produção em simultâneo, entre outros. Como estes métodos requerem uma grande quantidade de recursos informáticos (os quais necessitam de muito tempo para encontrar uma solução), podem não ser os mais indicados para aplicações em tempo real;
- **Métodos Heurísticos** — Alguns autores sugerem estes métodos de modo a minimizar o movimento de peças e a balancear as cargas em máquinas de tamanhos iguais. Outros focam-se na minimização do desequilíbrio de carga de trabalhos entre máquinas e na maximização taxa de transferência;
- **Métodos Baseados em Algoritmos Genéticos** — Estes métodos usam técnicas de procura estocástica que dependem do processo de seleção natural de modo a resolver problemas de maximização de taxa de transferência e de desequilíbrio do sistema;
- **Métodos de Simulação** — Quando um sistema não consegue ser avaliado analiticamente de modo a encontrar uma única solução, a simulação pode ser uma boa solução. Simulações em computador fornecem o comportamento completo e detalhado do sistema, o que permite encontrar a melhor solução. No entanto, devido à informação de que estes métodos necessitam para tomar uma decisão, tornam-se ineficazes para muitas aplicações em tempo real.

Para além dos problemas relativos ao carregamento de peças, há ainda a considerar os problemas com técnicas de agendamento, manuseamento de material, flexibilidade, conjunto máquina-ferramenta e técnicas de manutenção e controlo [8].

Tudo isto faz com que este tipo de sistemas seja de difícil implementação e que por isso nunca tenham tido muito sucesso.

2.3.2 Sistemas Reconfiguráveis de Manufatura

Um sistema reconfigurável de manufatura (SRM) é projetado com a finalidade de permitir alterações de estrutura (adição de máquinas, por exemplo), assim como de componentes de *software* e *hardware*, para que rapidamente possa adaptar as suas capacidades de produção e funcionalidades de uma família de peças a alterações súbitas de exigências regulamentares ou de mercado.

Um SRM possui cinco características chave:

- Modularidade — Todos os componentes do sistema devem ser projetados de forma modular;
- Integrabilidade — Tanto o sistema como os seus componentes devem ser projetados de modo a estarem prontos para uma rápida integração, assim como para à integração de nova tecnologia;

- Convertibilidade — Deve permitir uma troca rápida entre produtos existentes e uma rápida adaptação do sistema a produtos futuros;
- Diagnóstico — Deve identificar rapidamente a origem de problemas de qualidade e de fiabilidade que ocorram em sistemas grandes;
- Customização — A capacidade e flexibilidade do sistema deve ser projetada de acordo com a aplicação que irá ter [9].

Este tipo de sistemas combina a elevada taxa de transferência dos Sistemas Distribuídos de Manufatura (SDM) com a flexibilidade dos SFM, ao mesmo tempo que reage a mudanças de forma rápida e eficiente. Isto é conseguido devido à estrutura ajustável do sistema, permitindo a escalabilidade como resposta a exigências de mercado e adaptabilidade a novos produtos. As capacidades e funcionalidades de um SRM variam ao longo do tempo, permitindo flexibilidade não só na produção de uma variedade de partes, mas também na mudança do próprio sistema. O objetivo de um SRM é a utilização de um *design* do processo de manufatura que permita em simultâneo a reconfiguração do sistema inteiro, do *hardware* e do *software* de controlo [10].

2.3.3 Paradigmas Baseados em Sistemas Multiagente

A tecnologia baseada em agentes pode potencialmente resolver processos de decisão complexos, dinâmicos e distribuídos. Utilizando redes de elementos autónomos, mas ao mesmo tempo interativos, esta tecnologia oferece uma alternativa aos sistemas centralizados [11].

Um agente é um sistema computacional, situado num ambiente dinâmico e capaz de exibir comportamento autónomo e inteligente. Este pode ter um ambiente que inclua outros agentes. Esta comunidade de agentes interativos opera como um SMA.

Um agente possui várias características importantes:

- Atuam em nome do seu *designer* ou do utilizador que representam, de modo a atingir um objetivo específico;
- São autónomos, pois controlam simultaneamente o seu estado interior, assim como o seu comportamento no ambiente;
- Apresentam inteligência ao aplicarem regras fixas ao raciocínio, ao planeamento e às capacidades de aprendizagem;
- Interagem com o seu ambiente e com outros agentes;
- São adaptativos, ao serem capazes de ajustar o seu comportamento às mudanças no seu ambiente sem que seja necessária intervenção externa;
- Apresentam mobilidade ao conseguirem transportar-se para outros ambientes de modo a ter acesso a outros recursos ou agentes[12].

Estas características permitem ao agente interagir com o seu ambiente e mudar o

seu comportamento com base na sua experiência. Um SMA vê a cadeia de mantimentos composta por um conjunto de agentes inteligentes, cada um responsável por uma ou mais atividades na cadeia de mantimentos e cada um interagindo com outros agentes no planeamento e execução das suas responsabilidades. Como cada agente tem uma vista parcial do sistema, estes precisam de ter a capacidade de comunicar de modo a atingir um objetivo definido ou a resolver um problema. A interação entre agentes requer que estes se possam entender, utilizando uma linguagem de comunicação, ontologias e protocolos de interação [13].

No domínio dos agentes industriais, a FIPA (*Foundation for Intelligent Physical Agents*) estabeleceu uma série de especificações para o desenvolvimento de soluções baseadas em SMA. Alguns exemplos são as especificações da *FIPA Abstract Architecture*, que lidam com entidades abstratas que constroem serviços e ambientes para agentes, e as especificações da *FIPA Agent Communication* que lidam com a definição de ACL (*Agent Communication Language*) e protocolos de interação. A FIPA é o único *standard* para o desenvolvimento de SMAs e é normalmente adotada pelos programadores devido à elevada popularidade do JADE (*Java Agent Development Framework*) [14][15].

Ao longo das últimas duas décadas têm sido desenvolvidas e implementadas várias soluções baseadas em agentes na produção inteligente. O projeto MASCADA (*Manufacturing Control Systems Capable of Managing Production Change and Disturbances*) teve como objetivo o desenvolvimento de um sistema de controlo de manufatura baseado em agentes e de uma metodologia de projeto que permitisse aos programadores aplicar este algoritmo na vida real [16]. Mais recentemente foi desenvolvido o projeto IDEAS (Industrial Agents for the Fast Deployment of Evolvable Assembly Systems) [17], que visa a aplicação de características chave, em particular SMAs, de modo a permitir o funcionamento *instant/-plug and produce* de equipamento modular sem a necessidade de o reprogramar. Após, o projeto PRIME (Plug and PRoduce Intelligent Multi Agent Environment based on Standard Technology)[18] apresentou uma solução baseada em agentes, com especial foco na reconfigurabilidade e adaptabilidade da linha de produção. Desde então têm surgido cada vez mais projetos nesta área.

2.3.3.1 Sistemas Holónicos de Manufatura

O conceito Holónico surgiu dos trabalhos do autor e filósofo húngaro Arthur Koestler, que tentou entender o comportamento de sistemas complexos ao considerar as suas entidades constituintes como todos e como partes, ao mesmo tempo. De modo a descrever uma unidade básica de organização em sistemas biológicos e sociais, o autor inventou a palavra *holon*, que representa um *building block* autónomo e cooperativo de um sistema de manufatura para transformar, transportar, armazenar e validar informação e objetos físicos. Koestler propôs também o conceito de uma hierarquia aberta, sem limites ascendentes e descendentes, a qual é formada por *holons* e à qual deu o nome de *holarchy*, um

sistema de *holons* que cooperam de modo a atingir um objetivo. Assim, um Sistema Holónico de Manufatura (SHM) é uma *holarchy* que integra todo o espetro de atividades da manufatura, desde as encomendas, passando pelo *design*, produção e *marketing*, de modo a agilizar o processo de manufatura [19].

A ideia de utilizar este conceito em sistemas de manufatura surgiu por volta de 1990, inserida no programa SIM, como uma solução para lidar com as crescentes mudanças que estavam a afetar o setor da manufatura, ao utilizar os benefícios dos SHM, tais como a adaptabilidade, a estabilidade e o uso eficiente de recursos.

Este paradigma é normalmente implementado sobre os princípios dos SMA, havendo no entanto duas diferenças para um SMA:

- Um *holon* num SHM corresponde a um agente num SMA, diferindo na capacidade do *holon* de conter outros *holons*;
- Enquanto os agentes são entidades de *software*, os *holons* podem incluir partes de *software* e *hardware*.

Ao longo dos anos várias arquiteturas para sistemas deste tipo têm sido introduzidas, tais como a PROSA e a ADACOR.

A PROSA (Product Resource Order Staff Architecture) tem como objetivo providenciar as bases do *design* da arquitetura, definindo a terminologia e apresentando as entidades dos componentes e as suas responsabilidades. Esta arquitetura é composta por três *holons* principais (*product*, *resource* e *order*), que são responsáveis pelos aspetos do controlo de manufatura, planeamento de produtos e processos, recursos do *shop floor* e tarefas de manufatura, e por um quarto *holon* (*staff*), responsável por ajudar os outros *holons* com a sua visão global do sistema, oferecendo planos otimizados para situações específicas de manufatura. O projeto MASCADA, já referido anteriormente foi contruído tendo como base esta arquitetura [20].

A ADACOR (ADAptive holonic COntrol aRchitecture) tem como objetivo melhorar o desempenho dos sistemas de controlo de manufatura, ao melhorar a agilidade e a flexibilidade na reação à mudança. Esta arquitetura foca-se no *shop floor* e em SFM organizados em produção *job shop*, caracterizados por processos concorrentes e assíncronos com operações não preventivas e rotas alternativas. Esta arquitetura tem quatro classes de holons (*product*, *task*, *operational* e *supervisor*). As primeiras três são semelhantes às três principais da arquitetura PROSA, enquanto que a *supervisor* é diferente, pois introduz coordenação e otimização global no controlo descentralizado e é responsável pela formação e coordenação de grupos de *holons*. Cada *holon* possui apenas uma visão de parte do sistema, tendo que cooperar com os restantes para atingir o objetivo [21].

2.3.3.2 Sistemas Biónicos de Manufatura

Os Sistemas Biónicos de Manufatura (SBM) são baseados no funcionamento dos sistemas biológicos. Estes sistemas exibem autonomia e harmonia dentro de relações ordenadas hierarquicamente e têm propriedades muito semelhantes à operação de unidades de manufatura. Estas unidades obtêm informação do ambiente do *shop floor* e realizam operações. O resultado destas operações é enviado novamente para o ambiente. De modo a preservar a harmonia, coordenadores podem entrar em ação, tal como as enzimas nos sistemas biológicos. Sistemas reguladores, similares às hormonas, podem aplicar políticas e estratégias duradouras no ambiente de produção. As unidades de produção podem atuar de forma similar às células, formando estruturas de controlo hierárquicas. Nestas estruturas cada camada suporta e é suportada por camadas adjacentes. Assim, quando uma ordem é dada à camada superior, esta passa a informação até à mais inferior.

Os SBM operam com componentes distribuídos que comunicam entre si, dando informação das decisões que tomam. O conhecimento que os componentes têm dos processos é comparado à informação genética. Assim, os produtos possuem a informação de como uma peça deve ser produzida ou montada e comunicam com os recursos de modo a definir um plano de produção [22].

2.3.3.3 Sistemas Evolutivos de Produção

Os Sistemas Evolutivos de Produção (SEP) começaram com uma base em sistemas biológicos, tal como os SBM. No entanto, ao longo dos anos, os SBM foram-se desviando das suas origens, passando a sua implementação para sistemas hierárquicos. Apesar de algumas parecenças, estes dois tipos de paradigmas são diferentes e só com os SEP é possível atingir uma boa granularidade. O projeto IDEAS, já referido, utiliza como base este tipo de paradigma.

Os SEP focam-se nas mudanças nos sistemas de produção e na forma de gerir estas mudanças eficientemente. Este paradigma não tenta apresentar uma solução geral, que permita uma adaptabilidade geral capaz de incluir todo o tipo de produtos, mas sim uma abordagem mais específica, capaz de ser adotada por uma classe de produtos. O foco é a adaptação à mudança através da modularidade e da evolução passo a passo. A modularidade permite a evolução dos sistemas e impede que ao havendo uma melhoria numa parte do sistema, essa alteração comprometa a evolução do sistema como um todo.

A adaptabilidade autónoma deste tipo de sistemas é obtida através da evolução dos módulos do sistema através de propriedades como a auto-organização, o auto-diagnóstico e auto-reconfigurabilidade na reação às mudanças no ambiente. De modo a obter um sistema flexível, os sub-sistemas e os sistemas de controlo têm de ser extremamente adaptáveis. Quanto mais baixo um componente estiver posicionado na hierarquia, mais flexível tem de ser, de maneira a potenciar a flexibilidade do sistema [23].

2.4 Conclusões Gerais

É possível perceber pelos conceitos apresentados nesta secção que é necessário o aparecimento de um sistema de produção capaz de responder às atuais exigências da indústria. A descentralização, a reconfigurabilidade e a integrabilidade são algumas das características que os sistemas que têm aparecido recentemente tentam implementar. No entanto, verifica-se que estes novos paradigmas têm tido dificuldades na sua implementação devido à sua elevada complexidade. Assim, esta dissertação tenta explorar novas abordagens, de custo e complexidade mais reduzidos, os quais facilitem a implementação das ideias apresentadas por estes paradigmas.

Capítulo 3

Conceitos de Suporte

3.1 *Web Services* e Arquiteturas Orientadas a Serviços

Apesar de não existir uma definição aceite para o conceito de arquitetura orientada a serviços, uma definição geral pode ser "Uma Arquitetura Orientada a Serviços (AOS) é um conjunto de princípios arquitetónicos para à construção de sistemas autónomos e interoperáveis". Os sistemas autónomos são sistemas criados independentemente de outros, que operam independentemente do seu ambiente e que oferecem funcionalidades independentes. É favorecida a interoperabilidade ao abstrair-se a interface que um dispositivo expõe ao ambiente, da implementação desse mesmo serviço [24]. Numa arquitetura orientada a serviços não existem dependências diretas entre serviços, os quais estão estruturalmente desacoplados. É interoperável, na medida em que especifica uma interface que descreve os serviços por si oferecidos e os padrões de interação considerados. Os serviços são descritos de maneira a poderem ser descodificados por qualquer sistema. Estes serviços podem ser disponibilizados para uso geral [25].

De modo a implementar esta arquitetura, a tecnologia mais utilizada é a de *web services*. Como definido, "Um *web service* é um sistema de *software* desenhado para suportar interações interoperáveis de máquina para máquina através de uma rede. Tem uma interface descrita num formato processável por máquina, mais especificamente *Web Services Description Language* (WSDL). Outros sistemas interagem com um *web service* da forma estabelecida pela sua descrição, utilizando mensagens com um protocolo definido, normalmente transportadas utilizando *HyperText Transfer Protocol* (HTTP) com uma serialização *Extensible Markup Language* (XML) em conjunto com outros padrões relacionados com a *web*"[26]. Os dois protocolos de mensagens mais utilizados são o *Simple Project Access Protocol* (SOAP) e o *Representational State Transfer* (REST).

De forma a responder às exigências do mercado da manufatura no futuro (competição a nível global, a inovação e a introdução de novos produtos e mudanças frequentes nas exigências de mercado), será necessário que as fábricas e os sistemas de controlo do futuro tenham capacidades de integração dinâmica intra-empresa, cooperação inter-empresas, suporte de ambientes de *software* e *hardware* heterogéneos e ao mesmo tempo interoperáveis, agilidade através de adaptabilidade e reconfigurabilidade, possibilidade de adição de recursos sem perturbar operações e tolerância e recuperação de falhas.

Uma arquitetura orientada a serviços permite atingir muitos destes objetivos ao garantir:

- Capacidade de integração, ao permitir que serviços possam ser prontamente compostos com outros serviços, de modo a criar serviços de alto nível;
- Devido à abstração entre interface de serviços e implementação de serviços, os serviços podem ser materializados em plataformas heterogéneas de *software* e *hardware*;
- Melhoramento de agilidade, flexibilidade e adaptabilidade a mudanças, pois os serviços podem ser facilmente reconfigurados ou substituídos e as entidades de comunicação podem partilhar e trocar recursos e colaborar umas com as outras através de comunicação *peer-to-peer*;
- O custo de desenvolvimento é reduzido, pois a reutilização de serviços é facilitada e a programação de aplicações é feita ao mais alto nível de abstração;
- Como cada serviço inclui a sua pória complexidade, a escalabilidade torna-se numa característica inerente [27];

Um exemplo de um projeto onde esta tecnologia é aplicada é o *Service Infrastructure for Real-time Embedded Networked Applications* (SIRENA), que fornece uma infraestrutura de serviços para sistemas embutidos e dispositivos conectados através de várias redes em aplicações para indústria, casa, veículos e telecomunicações [28].

3.2 Internet of Things

A IoT tem sido um dos temas mais debatidos na área das tecnologias de informação e comunicação. O seu potencial para mudar as nossas rotinas diárias, assim como as suas variadas aplicações são algumas das razões pelas quais devemos olhar para este tema com muita atenção. De acordo com D. Mourtzis et al, "A IoT refere-se a uma rede de objetos interconectados que criam um ambiente inteligente" [29].

O seu objetivo passa por pegar em objetos do dia a dia e transmitir/ler informação a partir destes mesmos objetos, e interconectá-los, de maneira a que um computador possa ler, sentir, integrar, prever e reagir a todos os aspetos do mundo físico. Dispositivos, instrumentos e sensores irão ser amplamente utilizados em várias indústrias e irão ser

interconectados, fornecendo e consumindo informação disponível na rede. Podem obter a informação do ambiente através de sensores e podem ser controlados a partir da rede [30].

Atendendo ao objetivo desta tese, o foco será sobre as aplicações da IoT na manufatura, das quais as mais relevantes serão apresentadas de seguida.

- **Automação e Eficiência** — Nesta área é recolhida informação em tempo real do *shop floor*, sendo esta utilizada de forma a automatizar processos, mantendo e otimizando sistemas de produção e de design sem que seja necessária intervenção humana. Com informação recolhida em tempo real, algoritmos inteligentes e atuadores em rede, o *software* de controlo consegue automaticamente tomar decisões e controlar os atuadores de modo a diminuir os desvios do plano. Uma elevada quantidade de informação e algoritmos inteligentes permitem gerar soluções otimizadas. Utilizando *machine learning* consegue-se aumentar substancialmente o nível de autonomia de modo a controlar processos de produção e a lidar com várias discrepâncias;
- **Gestão de Energia** — Atendendo a que cerca de um terço da energia utilizada a nível global se deve à manufatura e tendo em conta os custos energéticos, este torna-se um tema bastante importante. A *IoT* pode ajudar a acompanhar continuamente o consumo de energia em tempo real ao colocar sensores em todos os locais pertinentes, sendo essa informação utilizada de modo a criar estratégias que permitam aumentar a eficiência energética;
- **Manutenção Pró-ativa** — Este tipo de manutenção é diferente de outros porque não espera que um problema ocorra para depois intervir. Neste caso é feita uma monitorização constante das condições das máquinas e dos equipamentos com o objetivo de detetar a origem de uma possível falha, podendo ser prontamente agendada uma intervenção que vise corrigir os valores anormais da origem dessa possível falha. Assim, a redução do tempo de inatividade é apenas uma consequência de uma estratégia que visa melhorar a saúde dos equipamentos durante o seu ciclo de vida, assegurando uma elevada produtividade, fiabilidade e robustez do sistema de produção enquanto que ao mesmo tempo diminui o número de intervenções de manutenção;
- **Gestão da Cadeia de Mantimentos** — Os sistemas que utilizam *IoT* conseguem ligar todas as partes da cadeia de mantimentos, partilhando informação em tempo real do *shop floor*, inventário, compras e vendas, manutenção e logística, de modo a que todas as partes tenham acesso à informação do fluxo de materiais e do tempo do ciclo de produção, identificando potenciais problemas antes de acontecerem e tomando medidas para os evitar;
- **Personalização em Massa** — As fábricas do futuro serão orientadas para a personalização em massa ao invés de produção em massa. A implementação de dispositivos

de *IoT* ao longo da cadeia de mantimentos permite não só reunir informação relativamente às preferências dos clientes mas também uma maior flexibilidade de integração de novas tecnologias ao nível da produção, facilitando a personalização dos produtos [31][32].

A conexão de objetos à *Internet* tem a capacidade de melhorar a sustentabilidade e a segurança das indústrias, permitindo uma interação eficiente entre o mundo físico e e a sua contra-parte digital, aquilo que normalmente é designado como um SCF.

Um sub-tema dentro da IoT, *Industrial Internet of Things* (IIoT), aborda o domínio das tecnologias de comunicação M2M com aplicações na automação. A IIoT é importante para um melhor entendimento do processo de manufatura, permitindo uma produção sustentável e eficiente. Este conceito, considerado como um dos pilares da manufatura inteligente, aborda a capacidade de conectar todos os recursos industriais, incluindo máquinas e sistemas de controlo, com os sistemas de informação e os processos de negócio. Como consequência, a grande quantidade de informação recolhida pode servir para alimentar soluções analíticas e levar a uma otimização das operações industriais. Por outro lado, a manufatura inteligente foca-se na etapa de manufatura dos produtos, como o objetivo de responder rapidamente e de forma dinâmica às mudanças na procura. Assim, a IIoT afeta toda a cadeia de valor e é um requisito para a manufatura inteligente [33].

3.3 Computação e Manufatura em Nuvem

O NIST (*National Institute of Standards and Technology*) define a computação em nuvem como "um modelo que permite o acesso de rede conveniente e *on-demand* a um conjunto partilhado de recursos de computação configuráveis (redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos ou removidos com o mínimo esforço de gestão ou interação com o fornecedor de serviços"[34]. Ou seja, a computação em nuvem permite mover todos os recursos computacionais que pessoas ou empresas têm instaladas nos seus computadores para à Internet. Deste modo é possível aceder a qualquer um destes recursos em qualquer lugar e a qualquer altura, bastando para isso ter acesso à Internet e a um computador. Este método não só é mais prático e flexível, pois os utilizadores não necessitam de instalar ou atualizar recursos e podem trocar, adicionar ou remover esses mesmos recursos na medida em que precisarem, assim como também é mais barato, pois elimina a necessidade de investimento em infra-estruturas complexas e em atualizações a equipamentos que se tornem obsoletos.

Na computação em nuvem, os recursos (servidores, armazenamento, *software*, etc...) são fornecidos como serviços e em três modelos diferentes:

- ***SaaS* (*Software* como um Serviço)** — Neste modelo, aplicações e *software* são corridos na nuvem, não sendo necessário para o utilizador instalar ou correr este tipo de

recursos no seu computador, baixando o investimento em *hardware* e em licenças de *software*;

- ***PaaS* (Plataforma como um Serviço)** — A plataforma é oferecida como um serviço, permitindo o desenvolvimento e implementação de aplicações, ao mesmo tempo que elimina os custos e a complexidade associados à compra e gestão das camadas inferiores de *hardware* e *software*;
- ***IaaS* (Infraestrutura como um Serviço)** — As capacidades de armazenamento e computação são fornecidas na forma de serviços assim como toda a estrutura necessária ao funcionamento dos modelos *PaaS* e *SaaS*.

Do conceito de computação em nuvem, nasce o conceito de manufatura em nuvem. Este é um modelo de manufatura orientado a serviços e suportado pela computação em nuvem, *IoT*, tecnologias de virtualização, tecnologias orientadas a serviços e tecnologias avançadas de computação. Tem como objetivo a partilha, circulação, elevada utilização e o uso *on-demand* de vários recursos e capacidades de manufatura ao fornecer serviços de manufatura seguros, baratos, fiáveis, *on-demand* e de elevada qualidade durante todo o ciclo de vida da manufatura. Num sistema deste tipo, vários recursos de manufatura podem ser conectados à Internet de forma inteligente, sendo controlados e geridos utilizando tecnologias de *IoT*. De seguida, estes recursos são virtualizados e agrupados em diferentes serviços de manufatura em nuvem, os quais podem ser obtidos, invocados, fornecidos e utilizados *on-demand* recorrendo para isso a tecnologias de virtualização, tecnologias orientadas a serviços e tecnologias de computação em nuvem.

As vantagens da manufatura em nuvem são similares às da computação em nuvem. Permite aumentar o tempo de utilização de recursos, pois estes podem ser utilizados por vários clientes. Elimina a necessidade de investimentos iniciais altos, pois não é necessário adquirir infraestruturas de elevado custo, sendo esta uma grande vantagem para pequenas e médias empresas. Permite também reduzir custos de administração, de energia e de manutenção. Facilita a capacidade das empresas em dimensionar a sua produção e negócios de acordo com a procura do cliente [35][36].

3.4 JADE

O JADE [14] é um ambiente de *software* que permite a criação de sistemas baseados em agentes de modo a gerir a informação dos recursos em rede de acordo com as especificações FIPA para SMA interoperáveis. Disponibiliza um *middleware* para o desenvolvimento e execução de aplicações baseadas em agentes, as quais podem trabalhar e interoperar num ambiente com fios ou *wireless*. Também apoia o desenvolvimento de SMA através de um modelo de agentes programáveis predefinidos e um conjunto de ferramentas de gestão e teste. É atualmente uma das *frameworks* mais utilizadas na indústria.

A *framework* é baseada num *middleware* que facilita o desenvolvimento de aplicações multiagente baseadas numa arquitetura de comunicação *peer-to-peer*. A informação, os recursos e o controlo podem ser totalmente distribuídos em terminais móveis, assim como em terminais na rede fixa.

O ambiente pode evoluir de forma dinâmica com agentes que aparecem e desaparecem do sistema de acordo com as necessidades e requisitos da produção. A comunicação entre agentes, independentemente de estes estarem a correr numa rede com fios ou *wireless*, é feita de forma simétrica, com um agente a poder iniciar a comunicação ou a responder a outro agente.

O JADE é totalmente desenvolvido em Java e é baseado nos seguintes princípios:

- **Interoperabilidade** — O JADE é compatível com as especificações FIPA. Assim, um agente JADE pode interoperar com outros agentes que não corram no ambiente JADE, desde que cumpram com o mesmo *standard*;
- **Uniformidade e portabilidade** — O JADE disponibiliza aplicações com um conjunto de APIs que são independentes da rede subjacente e da versão do Java;
- **Facilidade de utilização** — A complexidade do *middleware* está escondida por um conjunto de APIs simples e intuitivo;
- **Filosofia *Pay-as-you-go*** — Os programadores não necessitam de utilizar todos os recursos disponibilizados pelo *middleware*. Os programadores não têm de ter nenhum conhecimento sobre os recursos que não são utilizados, nem estes adicionam nenhuma sobrecarga computacional. O JADE inclui as bibliotecas de classes de Java necessárias para desenvolver agentes e para desenvolver o ambiente que disponibiliza os serviços básicos e que deve estar ativo num determinado dispositivo antes que um ou mais agentes possam ser executados nesse dispositivo. O conjunto de instâncias do ambiente JADE disponibiliza uma camada homogénea que esconde completamente das aplicações a complexidade e a diversidade das camadas inferiores (*hardware*, sistemas operativos, tipos de rede) [37].

3.5 FIPA

A FIPA [38] é uma organização com o objetivo de estabelecer especificações para tecnologias baseadas em agentes. Não promove tecnologia para um único domínio de aplicação, mas sim um conjunto de tecnologias para diferentes áreas de aplicação, as quais os programadores podem integrar de forma a criar sistemas complexos com um alto grau de interoperabilidade. Os primeiros documentos emitidos pela FIPA definem as normas gerais que permitem que uma sociedade de agentes exista, opere e seja gerida.

Foram definidos três papéis essenciais para o funcionamento de uma plataforma de agentes.

O *Agent Management System* (AMS) é o agente que supervisiona o acesso e uso da plataforma. É responsável por manter uma diretoria dos agentes residentes e por gerir o seu ciclo de vida.

O *Agent Communication Channel* (ACC) fornece um meio para o contacto básico entre agentes, dentro e fora da plataforma. É o método de comunicação padrão, oferecendo um serviço de encaminhamento de mensagens preciso e de confiança.

O *Directory Facilitator* (DF) é o agente que oferece um serviço de páginas amarelas à plataforma de agentes.

As especificações definem também a *Agent Communication Language* (ACL), utilizada pelos agentes para à troca de mensagens. É uma linguagem que descreve a codificação de mensagens e semântica, não especificando os mecanismos de transporte de mensagens [39].

Existem dois protocolos principais para à comunicação entre agentes, o FIPA Request e o FIPA ContractNet.

3.5.1 FIPA Request

O protocolo FIPA Request (figura 3.1) permite que um agente solicite a outro a execução de uma ação.

Para iniciar a comunicação, o agente iniciador envia um pedido ao agente participante. Este pode aceitar ou recusar o pedido. Caso seja aceite, o agente executa o pedido e informa o agente iniciador do resultado. Pode ser bem sucedido ou falhar. No caso de ser bem sucedido pode enviar uma mensagem simples que apenas diga que a execução foi concluída ou uma mensagem que informe que a execução foi concluída e quais os resultados [40].

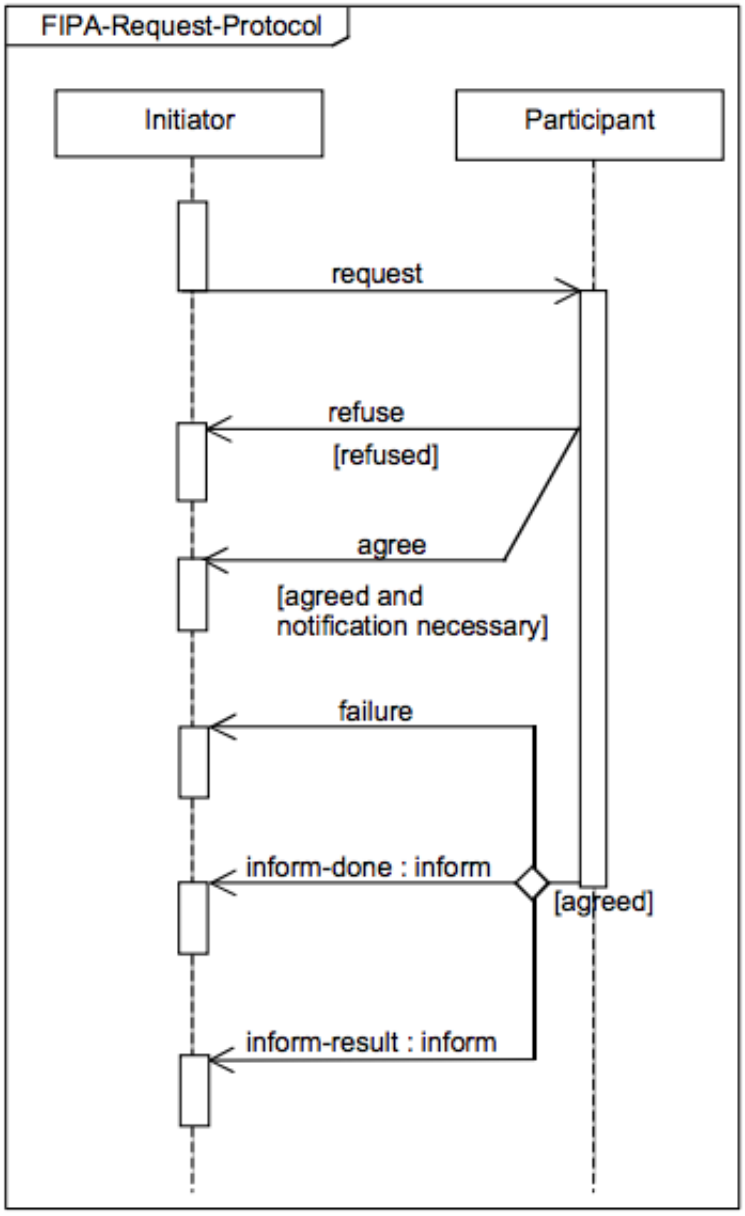


Figura 3.1: Protocolo FIPA Request [40].

3.5.2 FIPA ContractNet

O protocolo FIPA ContractNet (figura 3.2) permite que um agente que tenha uma determinada tarefa para executar, possa negociar a execução dessa tarefa com outros agentes.

O agente iniciador pede aos outros agentes participantes que façam uma proposta. Estes podem recusar ou enviar uma proposta. O agente iniciador decide qual a proposta que melhor se adequa às suas necessidades e aceita-a, recusando as restantes. O agente iniciador pode aceitar uma, mais que uma ou nenhuma proposta. Uma vez que a proposta é aceite, o agente participante executa a tarefa. Quando acaba, informa o iniciador do resultado da mesma. Tal como no protocolo FIPA Request, pode falhar, informar que a

tarefa está concluída ou informar que a tarefa está concluída e quais os seus resultados [41].

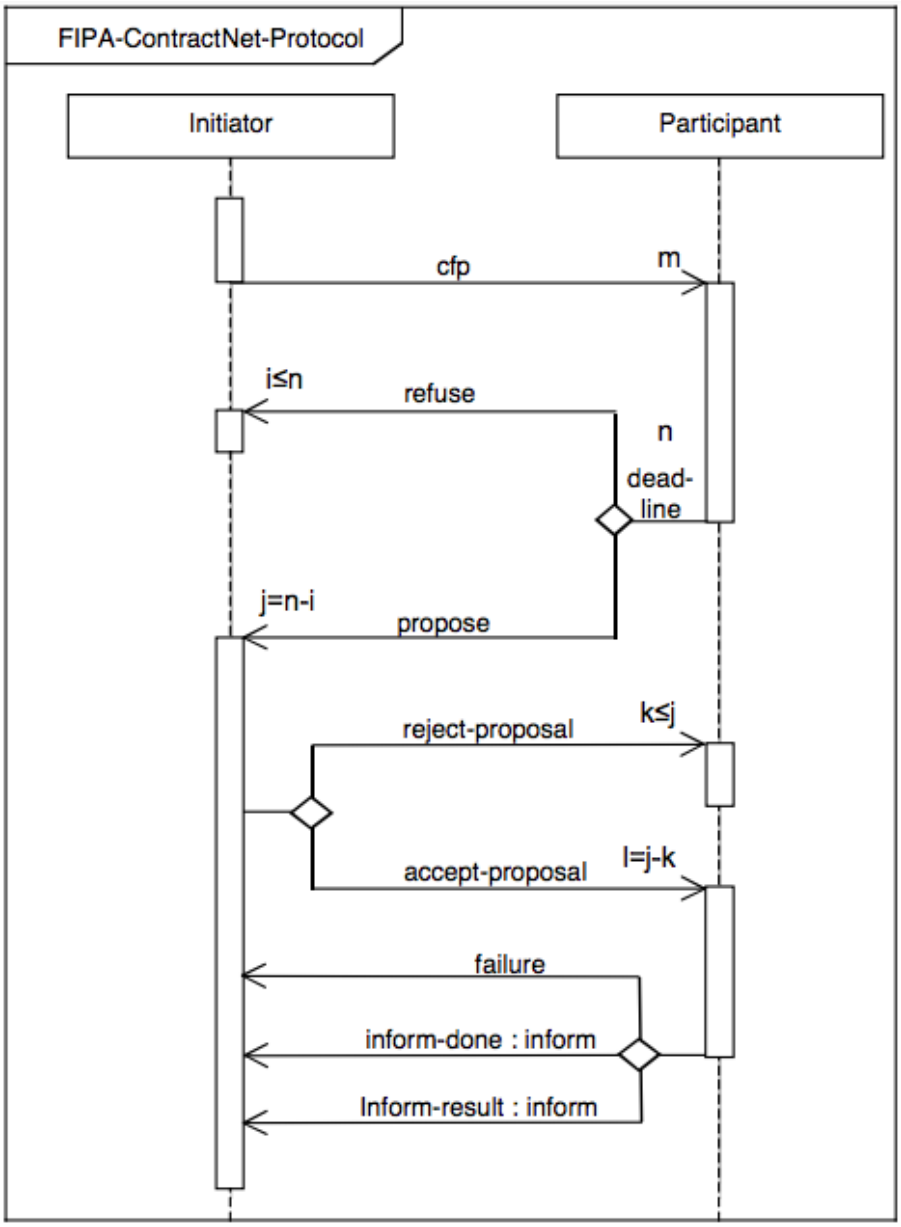


Figura 3.2: Protocolo FIPA ContractNet [41].

3.6 Protocolos de Transporte

A *Internet* tem dois principais protocolos na camada de transporte. Um protocolo sem conexão e outro orientado à conexão. O primeiro é o *User Datagram Protocol* (UDP) e a única coisa que faz é possibilitar o envio de pacotes entre aplicações. O segundo é o *Transmission Control Protocol* (TCP), este permite fazer conexões e adiciona fiabilidade com retransmissões, controlo de fluxo e controlo de congestão.

3.6.1 UDP

O UDP permite que aplicações enviem datagramas encapsulados sem terem que estabelecer uma conexão prévia. Transmite segmentos que consistem num cabeçalho de 8 bytes, seguido da informação. O cabeçalho contém a porta de origem e a porta de destino. Quando um pacote UDP chega, a sua informação é entregue ao processo anexado à porta de destino.

Uma área em que o UDP é especialmente útil é em situações cliente-servidor. O cliente envia um pequeno pedido ao servidor e espera uma pequena resposta em retorno. Caso o pedido ou a resposta sejam perdidos, o cliente pode simplesmente tentar de novo. Isto não só torna o código simples, como também requer um menor número de mensagens que um protocolo como o TCP [42].

3.6.2 TCP

O UDP é um protocolo simples com aplicações importantes, mas para à maioria das aplicações da *Internet*, é necessário uma entrega sequencial e fiável. Assim, existe o TCP.

O TCP foi especialmente desenvolvido de modo a oferecer um fluxo fiável de bytes ponto a ponto, através de uma inter-rede não fiável. Uma inter-rede difere de uma rede única, na medida em que pode ter diferentes topologias, larguras de banda, atrasos, tamanhos de pacote, entre outros parâmetros. O TCP foi desenvolvido para se adaptar de forma dinâmica às propriedades da inter-rede e de modo a ser robusto face a vários tipos de falhas.

O serviço TCP é obtido pelo remetente e pelo recetor ao criarem pontos finais de comunicação, chamados *sockets*. Cada *socket* tem um endereço que consiste no endereço IP (Internet Protocol) do anfitrião e um número de 16 bits local a esse anfitrião, chamado porta. Um *socket* pode ser utilizado para múltiplas conexões ao mesmo tempo, ou seja, duas ou mais conexões podem terminar no mesmo *socket*.

Uma das principais características do TCP é o facto de cada byte numa conexão TCP ter o seu próprio número de sequência de 32 bits. As entidades remetentes e recetoras trocam informação na forma de segmentos. Um segmento consiste num cabeçalho fixo de 20 bytes, seguido de zero ou mais bytes de informação [42].

3.7 Sockets

Tal como descrito anteriormente, um *socket* é um ponto final de comunicação, o qual tem um endereço anexado. A propriedade característica de um domínio é que os processos que comunicam no mesmo domínio têm o mesmo formato de endereço. Um *socket* comunica num único domínio.

Existem vários tipos de *sockets*, os quais representam classes de serviços. Cada tipo pode ou não ser implementado num qualquer domínio de comunicação. Se um tipo for implementado num determinado domínio, pode ser implementado por um ou mais protocolos desse domínio.

Existem dois principais tipos de *sockets*:

- ***Stream Sockets*** - Oferecem um fluxo de informação de dois sentidos, fiável e sequencial. Nenhuma informação é perdida ou duplicada na entrega. Este tipo é suportado no domínio da *Internet* pelo protocolo TCP;
- ***Datagram Sockets*** - Transferem mensagens de tamanho variável em ambas as direções. Não há garantia de que as mensagens sejam entregues pela mesma ordem em que foram enviadas, ou que não estejam duplicadas ou que não sejam entregues. Este tipo é suportado no domínio da *Internet* pelo protocolo UDP.

Quando a conexão para um *stream socket* é feita, os endereços do remetente e do recetor são conhecidos e não é necessário trocar mais informação relativa a endereços. Os dois podem depois trocar informação através desta conexão. Os *datagram sockets* não suportam conexões. Assim, trocam datagramas, os quais devem ser endereçados individualmente [43].

Capítulo 4

Arquitetura

Tal como referido anteriormente, a necessidade de evolução dos sistemas de manufatura fez com que vários paradigmas de produção (SFM, SRM, SMA, entre outros) aparecessem ao longo dos anos. No entanto a sua implementação é dificultada por restrições de *hardware*, o qual não consegue suportar a camada de controlo.

Assim, esta dissertação propõe uma arquitetura dividida em três camadas: Recursos Físicos, Dispositivos IoT e SMA. Com isto, pretende-se alocar ao dispositivo IoT o controlo das tarefas que são mais críticas em termos de tempo, como são o controlo de execução de habilidades num recurso físico em tempo real, enquanto que o resto do controlo é alocado ao SMA, o qual é corrido numa máquina ou conjunto de máquinas com maior poder computacional.

A utilização de um SMA maximiza a capacidade do sistema, oferecendo autonomia às suas entidades e tornando-as capazes de ajustar o seu comportamento a mudanças na sua execução. Ao juntar um dispositivo IoT de baixo custo capaz de controlar um recurso físico e capaz de comunicar com o SMA, torna possível a fácil integração de módulos no sistema, agindo como um facilitador da característica *Plug&Produce*.

4.1 Arquitetura do Sistema

Tal como referido, a arquitetura proposta foca-se na separação do controlo entre dispositivo IoT e SMA. Uma vez que o SMA é quem controla a sequência de passos do sistema necessária à produção de um produto, pois é este que conhece o seu plano de execução, isto permite que o dispositivo IoT utilizado para controlar o recurso físico e para comunicar com o SMA seja mais simples. Com isto, permite a utilização de uma maior variedade

de dispositivos capazes de oferecer estas características, baixando o custo do sistema e tornando o mesmo mais acessível.

O SMA abstrai produtos e recursos e é responsável pela gestão do sistema. O dispositivo IoT é responsável pelo controlo de execução de uma habilidade, pois apenas ele interage com o recurso físico. Assim, quando um utilizador lança um produto no sistema, este decide, habilidade a habilidade, qual o melhor recurso a utilizar para à sua execução (figura 4.1).

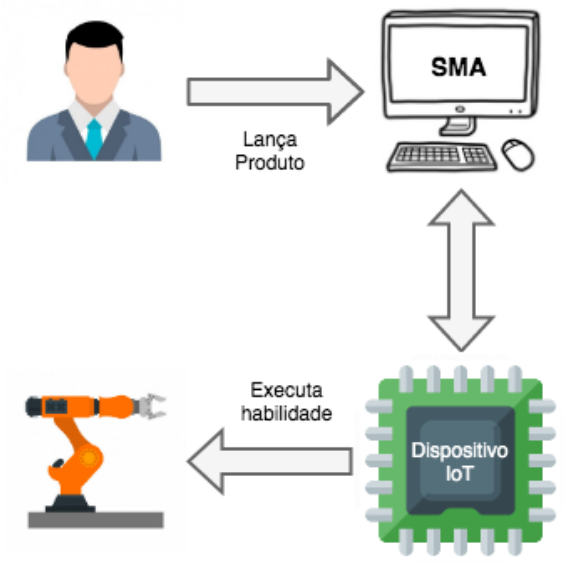


Figura 4.1: Arquitetura simples do sistema.

É também importante clarificar o conceito de habilidade. Assim, neste sistema, as habilidades estão diretamente associadas às capacidades ou funcionalidades dos recursos fisícos. Estas podem representar uma funcionalidade do recurso ou um conjunto delas.

Cada recurso pode oferecer uma ou mais habilidades, assim como, dois recursos diferentes podem oferecer as mesmas habilidades. É o SMA que gere e escolhe qual o melhor recurso a utilizar para cada produto. Isto é possível devido à modularidade do sistema proposto, em que cada recurso é um módulo independente, permitindo assim essa gestão por parte do SMA. O número de recursos possíveis a utilizar apenas é limitado pelo espaço físico do *shop-floor*. Mais recursos oferecem mais soluções e permitem uma gestão mais eficiente por parte do SMA.

No entanto, o foco desta solução recai na capacidade de integrar e remover módulos do sistema com facilidade, sem interromper o processo de produção. O objetivo passa por utilizar dispositivos baseados em IoT como facilitadores da característica *Plug&Produce*.

Um módulo é composto por um dispositivo IoT e por um ou mais recursos físicos ligados ao mesmo, desde que todos os recursos físicos ligados estejam disponíveis na

mesma estação. Assim, pode ser combinado mais que um recurso físico de modo a oferecer um maior leque de habilidades. Cada dispositivo é responsável por informar o sistema das habilidades que pode executar e da sua localização e apenas tem que passar estas informações ao sistema aquando da sua integração. Por sua vez, o sistema tem que ser capaz de detetar a adição e remoção de módulos.

Para um melhor entendimento deste processo, considere-se o caso seguinte:

O SMA é iniciado na máquina sem ter qualquer módulo disponível. O mesmo fica à espera que um módulo se tente ligar (figura 4.2).

Figura 4.2: Estado 1.

De seguida, um utilizador liga um módulo à alimentação e em rede com a máquina que corre o SMA, sendo o mesmo composto pelo dispositivo IoT e por um recurso físico. Este módulo, o qual oferece um conjunto de habilidades, é integrado no sistema, ficando disponível para ser utilizado (figura 4.3).

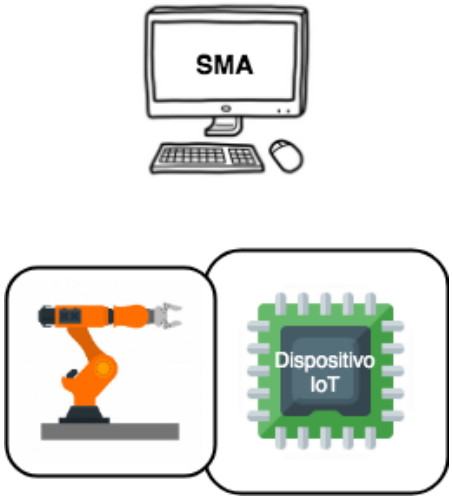


Figura 4.3: Estado 2.

É posteriormente ligado um novo módulo. Este é composto por um dispositivo IoT e dois recursos. Este módulo, que oferece as mesmas habilidades que o módulo anterior e algumas adicionais, é também integrado no sistema, passando a existir dois módulos disponíveis (figura 4.4).

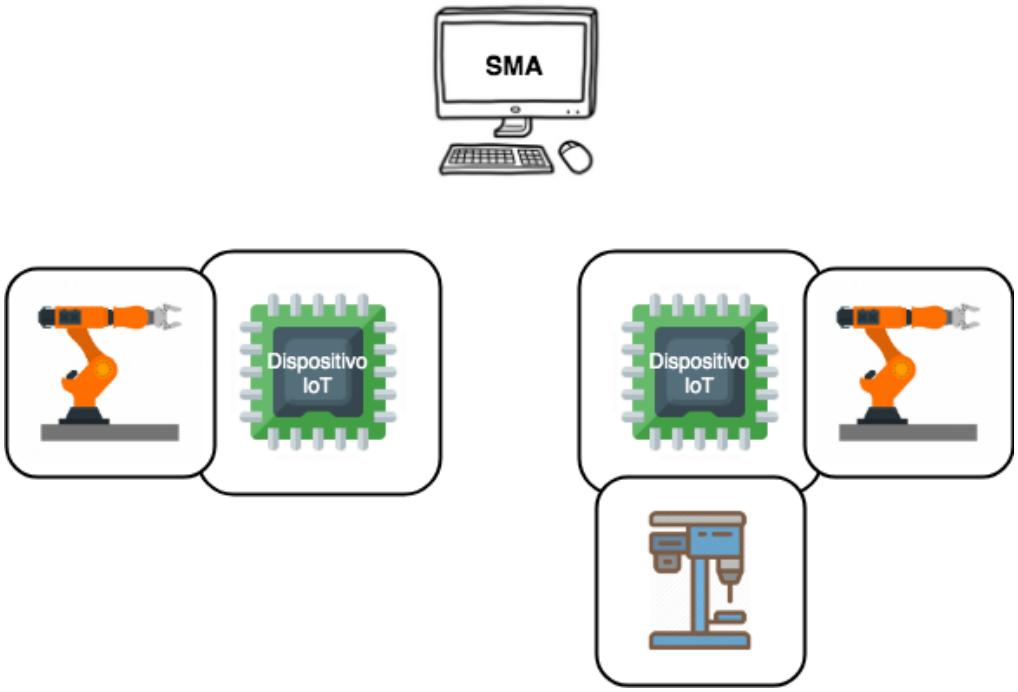


Figura 4.4: Estado 3.

Finalmente um utilizador desliga o dispositivo IoT do primeiro módulo, sendo este removido do sistema, o qual passa a ter apenas um módulo disponível (figura 4.5).

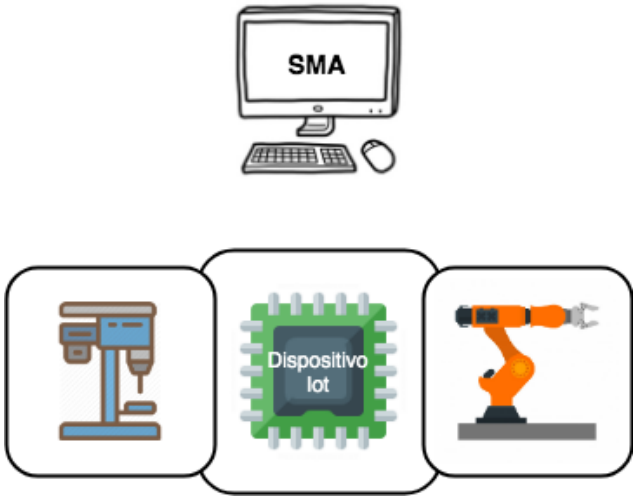


Figura 4.5: Estado 4.

Assim, a chave para atingir esta capacidade, reside na comunicação entre SMA e dispositivo IoT. De modo a atingir uma arquitetura escalável e integrável, é necessário que a interface entre SMA e dispositivos seja genérica. A máquina que corre o SMA e os dispositivos são ligados numa rede local através de *Ethernet*, sendo esta a base para a sua comunicação.

A figura 4.6 representa a arquitetura completa do sistema. Um utilizador pode lançar um ou mais produtos no sistema, sendo que cada um tem um plano de execução diferente. O SMA atribui a cada habilidade o recurso necessário à sua execução. Os dispositivos IoT quando recebem o pedido para executar uma determinada tarefa, interagem com o recurso físico de modo a executar a mesma e informam o SMA quando esta é terminada.

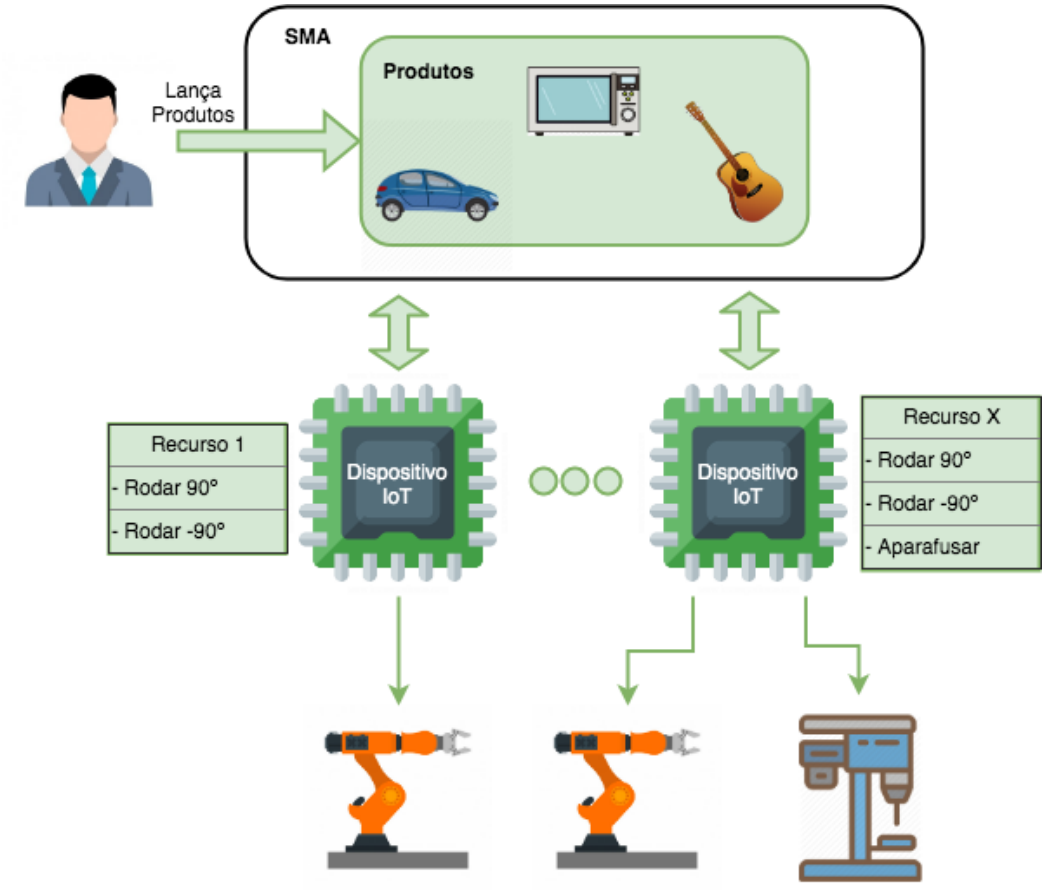


Figura 4.6: Arquitetura completa do sistema.

Resumindo, ao utilizar o SMA é possível atingir um sistema reconfigurável e de fácil gestão. No entanto, a mais valia desta arquitetura passa pela distribuição do controlo do sistema entre SMA e dispositivo IoT. Uma vez que todos os agentes correm na máquina, isto permite a simplificação dos dispositivos que fazem a ponte entre o SMA e os recursos físicos. Estes apenas terão que conseguir comunicar com o SMA de modo a permitir a sua adição e remoção, bem como a execução de habilidades, permitindo assim atingir um sistema escalável e integrável.

4.2 Arquitetura do Sistema Multiagente

A arquitetura proposta foi baseada no projeto IDEAS[17], sendo no entanto mais simples, pois o foco deste projeto é a utilização de dispositivos IoT como facilitadores da característica *Plug&Produce* e não o SMA. Cada um dos componentes do sistema está abstraído

por um agente. Sistema de transporte, produto e recurso são abstraídos respetivamente por, TA, PA e RA. Existem ainda mais dois agentes de suporte, DF e DA. Na figura 4.7 é apresentada uma ilustração do funcionamento do SMA.

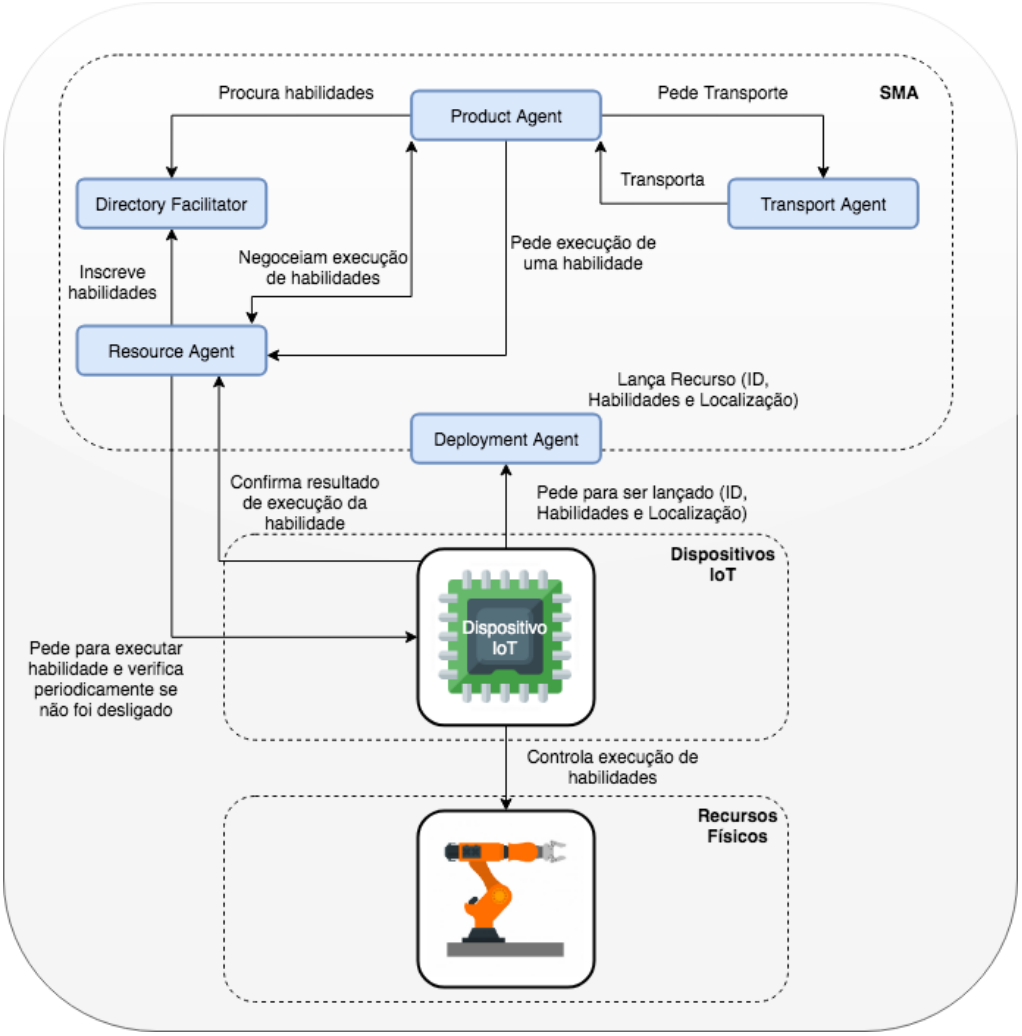


Figura 4.7: Arquitetura do SMA.

Nesta solução, todos os agentes correm numa máquina, sendo que o dispositivo de controlo apenas é requisitado para executar uma habilidade.

A tabela 4.1 apresenta a função de cada agente no sistema.

Nas subsecções seguintes será explicado o funcionamento dos agentes mais importantes para à solução. O PA, que controla todo o plano de execução de um produto, e DA e RA, que interagem com o dispositivo de controlo, possibilitando a integração de módulos e a execução de habilidades pelo recurso físico. O agente TA não foi desenvolvido no âmbito desta tese, tendo sido utilizado um já existente.

Tabela 4.1: Descrição dos Agentes.

Agente	Função
Directory Facilitator (DF)	Funciona como um serviço de páginas amarelas. Nele estão registadas as habilidades que os agentes oferecem. Pode ser consultado por agentes para procurar outros agentes e habilidades.
Deployment Agent (DA)	Lança agentes do tipo RA a pedido do dispositivo de controlo. Recebe o *ID*, a posição e as habilidades do recurso e lança-o no sistema com essas informações.
Product Agent (PA)	É o nível mais alto de abstração do sistema. Responsável por todo o plano de execução de um produto. Requisita e decide quais os recursos a utilizar na execução de um produto.
Resource Agent (RA)	O RA é o nível mais baixo de abstração do sistema. Interage com o dispositivo de controlo do recurso físico e apenas pede para que execute uma determinada habilidade. É responsável pela execução de habilidades e de informar o PA da localização física onde as suas habilidades estão disponíveis. Controla também a remoção física de um recurso, eliminando-se a si próprio do sistema quando tal acontece.
Transport Agent (TA)	É requisitado por agentes do tipo PA para transportar o produto entre estações de acoplamento.

4.2.1 *Product Agent* (PA)

Como já referido anteriormente, este agente é o nível mais alto de abstração do sistema. Tem o plano de execução do produto e é responsável por coordenar com os restantes recursos o seu seguimento. Na figura 4.8 apresenta-se um fluxograma do seu funcionamento.

Quando um agente deste tipo é criado, o seu primeiro passo é registar-se no DF. Após este passo, ao seguir o plano de execução, o agente percorre as habilidades e negoceia uma a uma, procurando um recurso capaz de a produzir e executando-a de seguida. O processo de negociar uma habilidade consiste em procurar no sistema todos os RA capazes de executar essa habilidade, escolhendo o que tenha o custo mais baixo. Caso esta negociação falhe, o agente remove-se do sistema.

De seguida, o RA escolhido informa o PA da sua localização, o qual, por sua vez, pede ao TA que o transporte para essa estação. Caso este processo falhe, o agente remove-se do sistema. Chegando à estação, o RA executa a habilidade negociada. Caso falhe, o agente remove-se do sistema. Depois de executada a habilidade, o PA verifica se existem mais habilidades a executar e caso haja, repete o processo anterior. Se já tiver executado todas as habilidades, o agente remove-se do sistema.

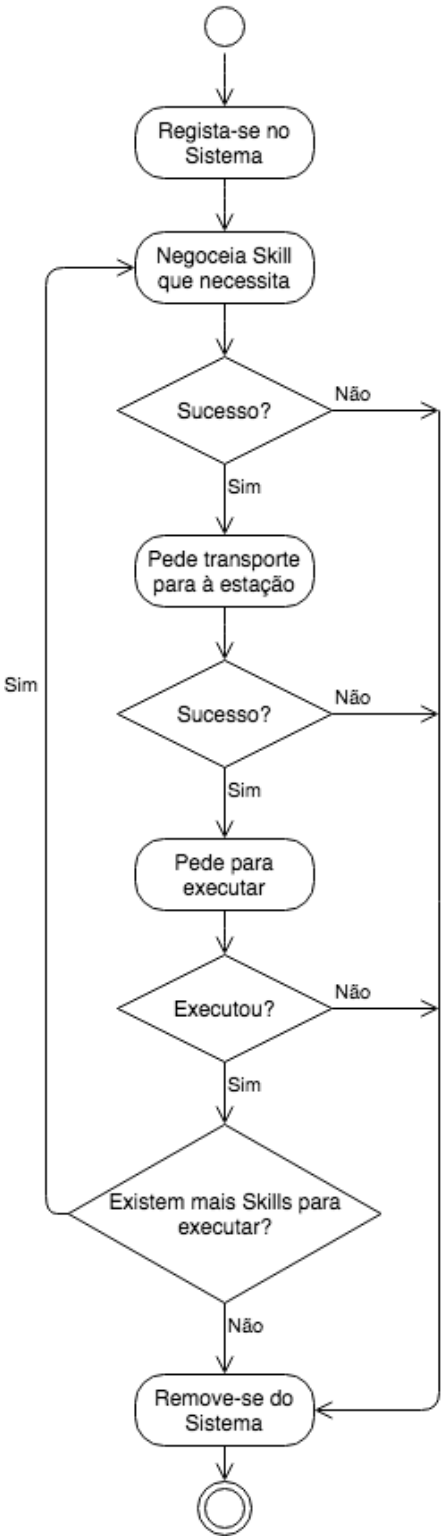


Figura 4.8: Fluxograma do PA.

4.2.2 *Deployment Agent* (DA)

Este agente (figura 4.9) faz a conexão entre módulos de produção e SMA. É responsável por lançar agentes do tipo RA no sistema. Inicia-se quando o sistema é ligado e fica à espera de receber um pedido de ligação. Quando um dispositivo IoT é ligado, este envia um pedido ao DA para que o lance no sistema. Este pedido contém a identificação do dispositivo, a informação das habilidades que o recurso oferece, assim como a sua localização.

De seguida o DA tenta lançar um RA com estas informações. Caso seja bem sucedido, fica novamente a aguardar por pedidos de ligação. Caso falhe, gera a mensagem de erro e aguarda por pedidos de ligação. Este agente apenas é desligado quando o sistema é desligado, terminando assim o seu processo.

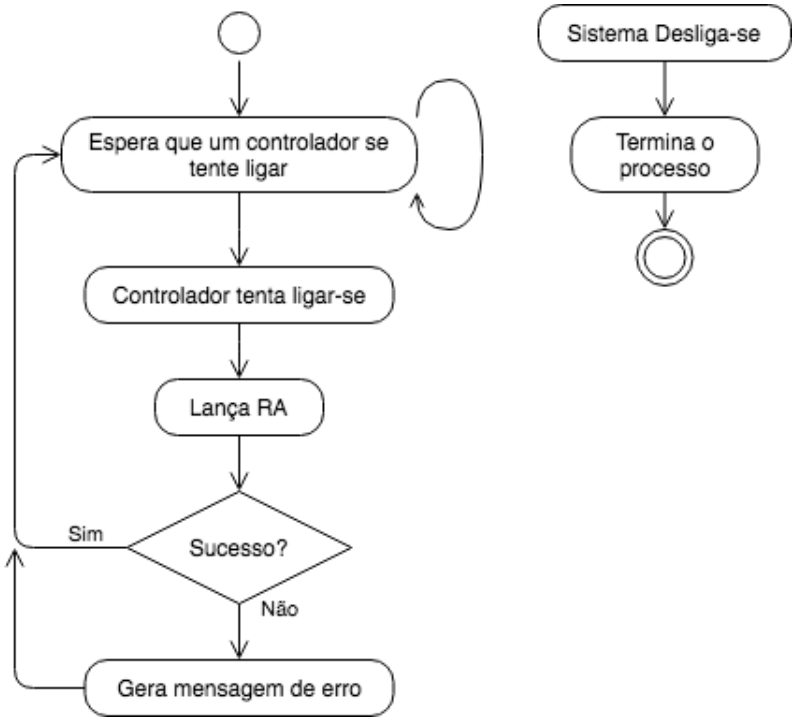


Figura 4.9: Fluxograma do DA.

4.2.3 *Resource Agent* (RA)

Sendo o agente de mais baixo nível, este é requisitado pelo PA para que execute uma determinada habilidade. Na figura 4.10 é apresentado o fluxograma do funcionamento deste agente.

Quando é lançado, este agente regista-se no DF, registando a sua localização e as habilidades que oferece. De seguida, inicia um processo periódico que visa verificar se o módulo de produção que abstrai ainda se encontra ligado. Caso se desligue, este RA desinscreve-se do DF e remove-se do sistema.

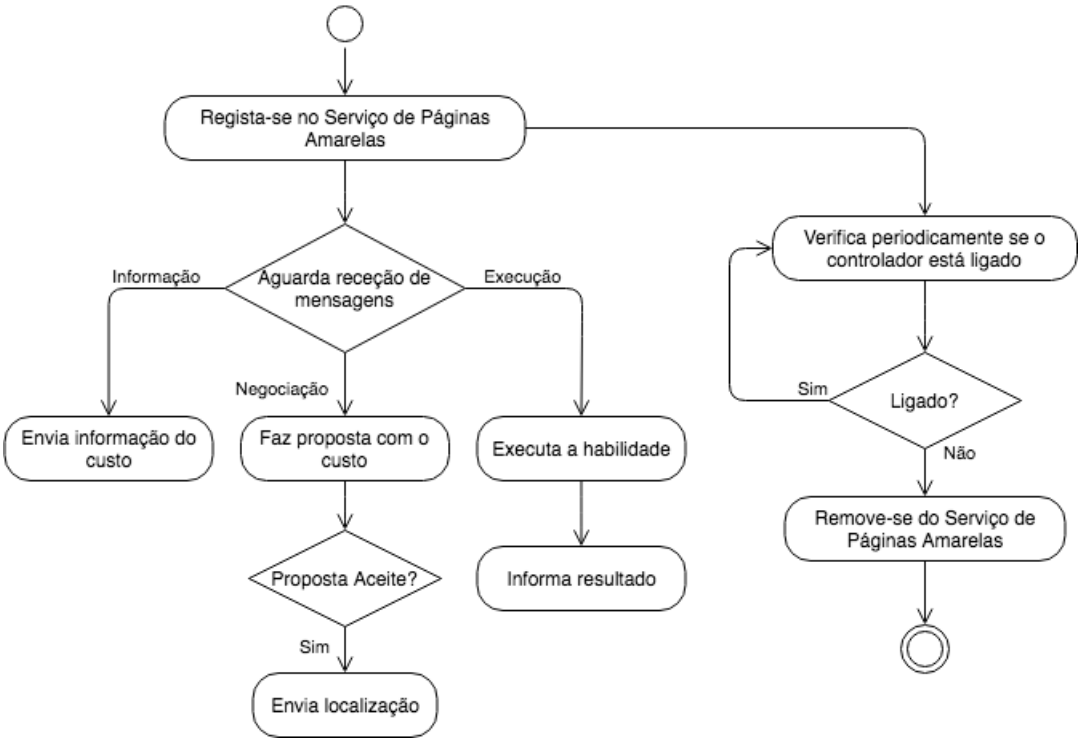


Figura 4.10: Fluxograma do RA.

Ao mesmo tempo, espera pela receção de mensagens do PA. As mensagens podem ser pedidos de informação, de negociação ou de execução. Caso seja um pedido de informação, este agente responde com o seu custo. Se for um pedido de negociação, este faz uma proposta com o seu custo. Se a proposta for aceite este envia a sua localização. Por último, caso seja um pedido para executar uma habilidade, este interage com o dispositivo de controlo para que execute a habilidade e informa o PA do resultado da mesma.

4.3 Dispositivo IoT

O dispositivo IoT faz a ponte entre SMA e recurso físico, tendo por isso que ter a capacidade de interagir com ambos. Possui a informação de que habilidades os recursos físicos ligados a si podem executar, bem como da sua localização no *shop-floor*. De forma a permitir a modularidade da arquitetura, o dispositivo está ligado fisicamente a um ou mais recursos físicos, formando assim um módulo (figura 4.11). Cada módulo é um recurso capaz de executar uma ou mais habilidades.

Quando é ligado, tem de passar a sua identificação, informação das habilidades que oferece e a sua localização ao SMA. Torna-se assim muito importante o modo de comunicação com o SMA, não só na sua integração no sistema, mas também quando receber pedidos do SMA para que execute habilidades. Terá de ser rápido e eficiente para não interromper o processo de produção. De seguida, passa a funcionar como um prestador

de serviços, sendo que não corre nenhum tipo de agente, não necessitando assim de um poder de processamento elevado. É no entanto essencial que este dispositivo tenha várias saídas e entradas, de modo a ativar/desativar atuadores e a ler sensores, controlando assim a execução de uma habilidade pelo recurso físico.

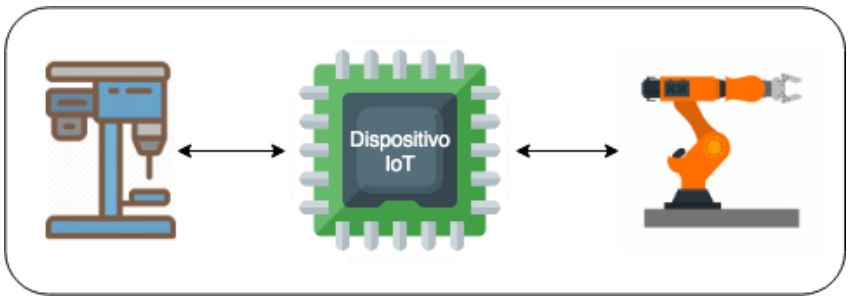


Figura 4.11: Exemplo de um módulo de produção com dois recursos físicos.

Nesta arquitetura, os dispositivos são ligados numa rede local com a máquina que corre o SMA através de Ethernet. O SMA terá ainda de saber quando um recurso é desligado para proceder à remoção do mesmo do sistema.

Os processos de integração e remoção de um módulo do sistema, bem como o processo de execução de uma habilidade são explicados com maior detalhe nas secções seguintes.

4.3.1 Integração de um dispositivo

Uma vez ligado em rede com a máquina que corre o SMA, o dispositivo terá de ser capaz de se integrar no sistema assim que é ligado à alimentação. Assim, uma vez ligado, o primeiro passo passa por enviar ao SMA as informações necessárias à utilização do módulo. É enviada a identificação, o conjunto de habilidades que o módulo oferece e a sua localização no *shop-floor*.

Quando envia esta informação, o dispositivo fica à espera de receber a confirmação do SMA de que foi lançado no sistema, passando depois a atuar como um prestador de serviços, esperando por pedidos do SMA.

Este processo só termina quando o dispositivo é desligado da alimentação. Uma ilustração deste processo é apresentada na figura 4.12.

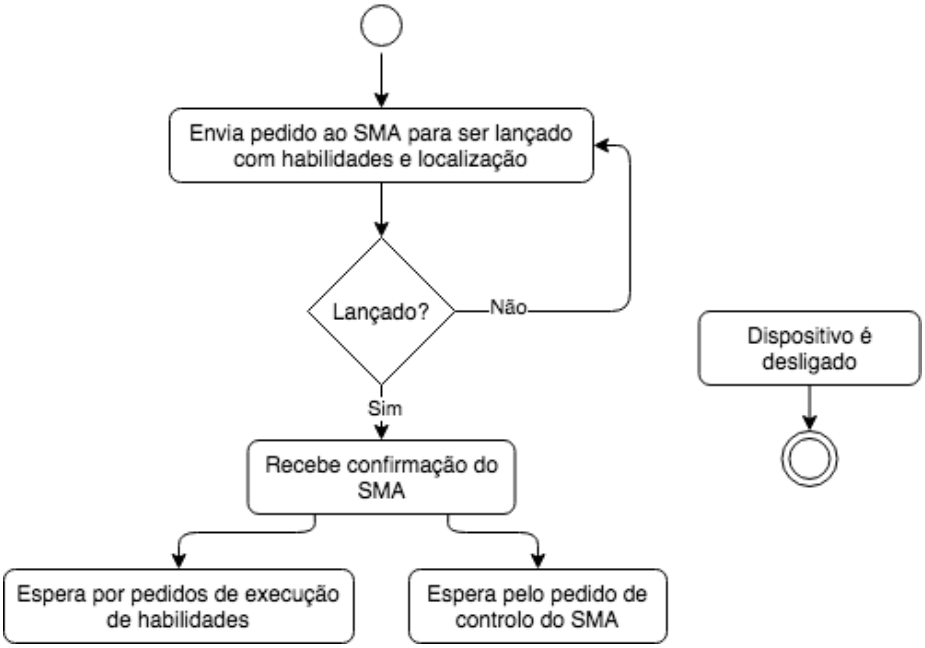


Figura 4.12: Fluxograma da integração de um dispositivo.

4.3.2 Execução de uma habilidade

Depois de integrado no sistema, o dispositivo espera por pedidos por parte do SMA. No caso do pedido ser para que execute uma habilidade, o dispositivo terá que interagir com um ou mais recursos físicos ligados a ele, para que passo a passo, a habilidade seja executada. Esta interação é feita através da ativação e desativação de atuadores e da leitura de sensores.

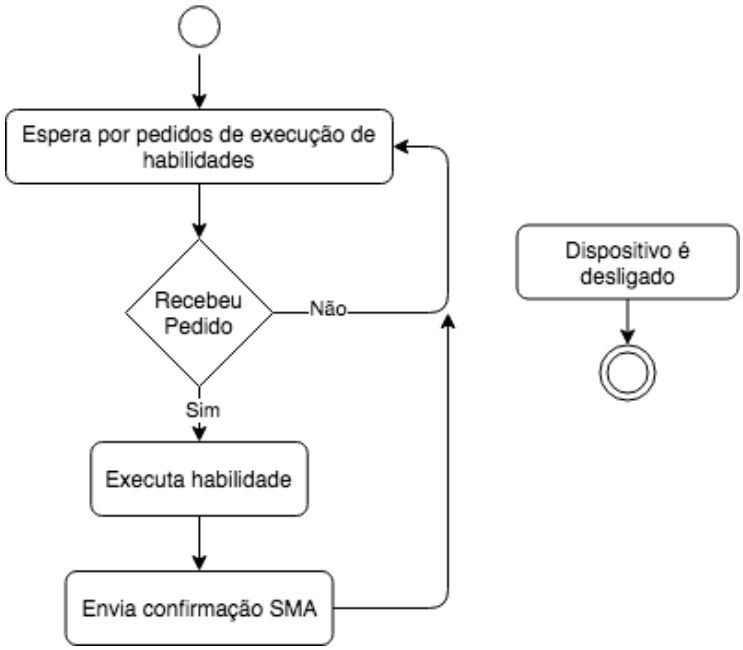


Figura 4.13: Fluxograma da execução de uma habilidade no dispositivo.

Após terminar a execução da habilidade, o dispositivo envia uma mensagem de confirmação ao SMA, ficando novamente à espera de pedidos do SMA.

Tal como no caso anterior, este processo, apresentado na figura 4.13, só termina quando o dispositivo é desligado da alimentação.

4.3.3 Remoção de um dispositivo

A remoção de um dispositivo do sistema é feita em dois passos. O primeiro acontece quando um utilizador desligar o dispositivo da alimentação. No entanto, para que um módulo não fique disponível no sistema, sem que na realidade o esteja, o SMA terá de ser capaz de identificar quando um dispositivo é desligado para o remover também do sistema, sendo este o segundo passo.

Assim, existe um processo (figura 4.14) que é executado periodicamente para controlar a remoção de um dispositivo. Ao mesmo tempo que espera pelos pedidos de execução de habilidades, espera também por pedidos de controlo por parte do SMA. Ao receber um pedido deste tipo, o dispositivo envia uma mensagem, confirmando que se encontra ligado.

É quando este controlo falha que o SMA sabe que o dispositivo já não está disponível para ser utilizado e procede à sua remoção do sistema. A remoção deste dispositivo da alimentação faz com que o processo acabe, causando a falha no controlo, sendo o dispositivo removido do sistema.

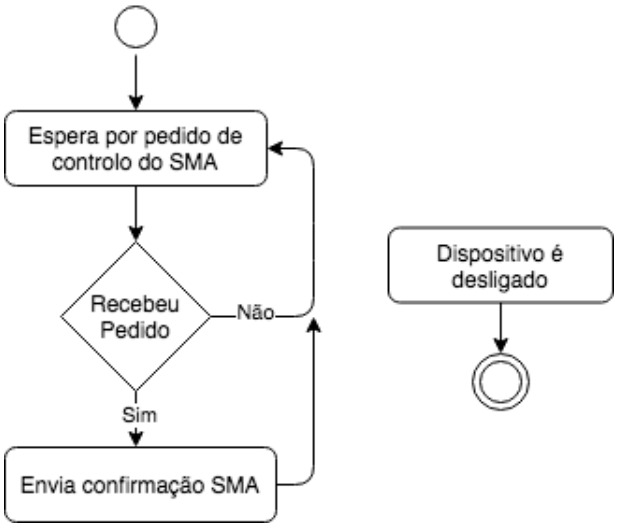


Figura 4.14: Fluxograma do controlo feito ao dispositivo.

Com esta capacidade de controlar a execução de habilidades e de comunicar com o SMA, torna-se então possível utilizar o dispositivo IoT como um facilitador da característica *Plug&Produce*.

Capítulo 5

Implementação

Tal como referido, o SMA foi alocado a uma máquina, tendo sido implementado em JAVA e utilizando o ambiente de desenvolvimento JADE [14].

Para os dispositivos baseados em IoT foram escolhidos dispositivos baseados em Arduino, os quais permitem controlar os sensores e atuadores dos recursos físicos de modo a controlar a execução de habilidades. Estes suportam ainda a ligação Ethernet necessária para comunicarem com o SMA, o qual corre fora destes dispositivos.

A comunicação entre SMA e dispositivo IoT, a qual permite a adição e remoção de módulos, bem como a execução de habilidades, foi implementada recorrendo a *sockets*.

A figura 5.1 apresenta uma visão geral da implementação escolhida.

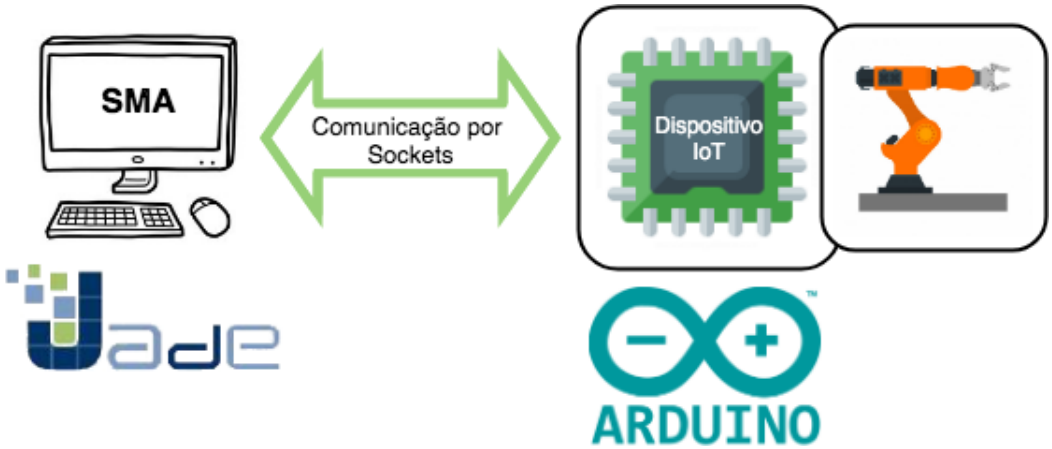


Figura 5.1: Visão geral da implementação.

De seguida é apresentada a explicação mais detalhada desta implementação.

5.1 Sistema Multiagente

5.1.1 *Directory Facilitator*

O serviço de páginas amarelas foi implementado utilizando o DF, disponibilizado pelo JADE. Os agentes podem registar-se no DF com um identificador próprio e podem também registar que serviços oferecem. Assim, a qualquer momento, um agente pode procurar no DF por outro agente que ofereça o serviço que este procura.

5.1.2 Classes Auxiliares

Foram utilizadas duas classes auxiliares, as quais permitem evitar a repetição de código.

A classe *Constants* (fig. 5.2) não tem nenhuma função, contém apenas um conjunto de constantes que são utilizadas nas restantes classes. Estas constantes simplificam a alteração de ontologias ou de temporizadores, por exemplo.

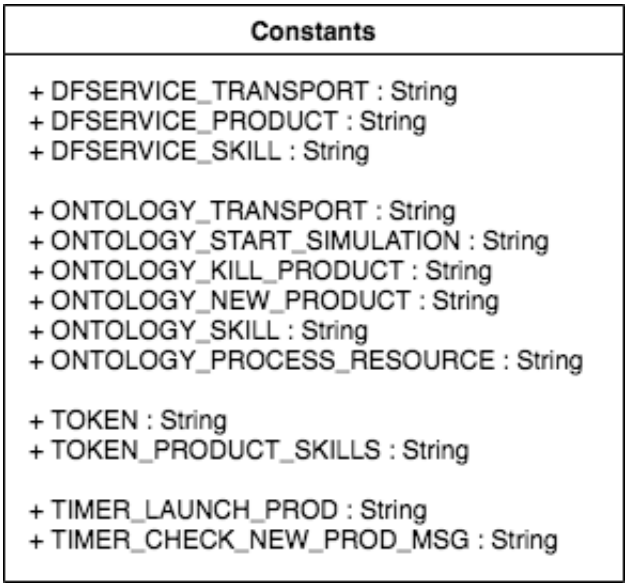


Figura 5.2: Classe *Constants*

Foi também criada a classe *DFInteraction*, apresentada na figura 5.3. Esta, tal como o nome indica, contém funções que são utilizadas de forma recorrente por outras classes e que possibilitam a interação com o DF.

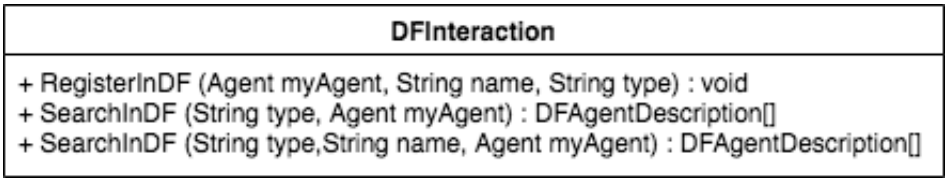


Figura 5.3: Classe *DFInteraction*

5.1.3 *Product Agent*

A classe responsável pela implementação do PA é apresentada na figura 5.4. Esta classe *extends* a classe Agent do JADE.

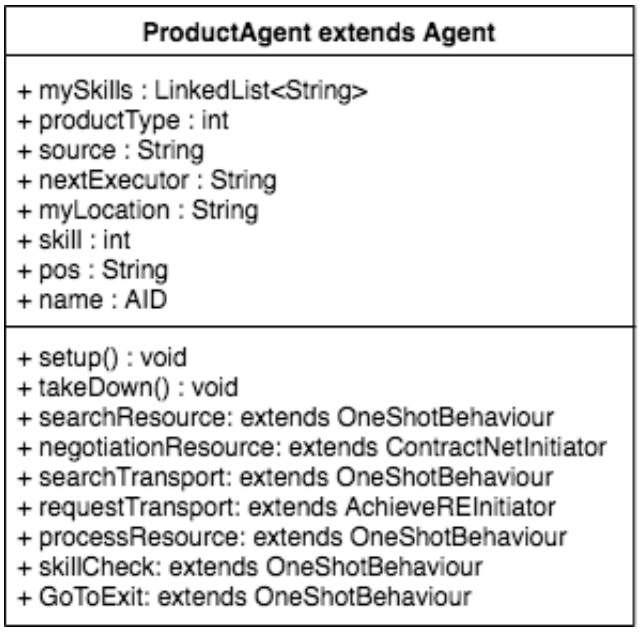


Figura 5.4: Classe *ProductAgent*

A função setup é responsável por inscrever o PA no DF. De seguida existem uma série de subclasses responsáveis por executar o plano de produção.

O searchResource procura no DF todos os recursos que oferecem a primeira habilidade do plano.

De seguida, no negotiationResource, negoceia a execução da habilidade com cada um dos recursos encontrados anteriormente. Depois de receber todas as propostas, avalia qual a que tem o menor custo e aceita-a. Ao aceitar a proposta, o recurso envia a sua localização para que o PA possa requisitar o transporte.

No searchTransport, o PA procura um TA que o transporte para a posição recebida. De seguida, no requestTransport, pede ao TA encontrado que proceda ao transporte e espera que este o informe de que chegou à posição.

Ao chegar à posição executa o processResource, responsável por processar o recurso escolhido. Este executa o requestResource, pedindo ao RA que execute a habilidade pretendida.

Depois de finalizada a execução da habilidade, executa o skillCheck, que verifica se existem mais habilidades no plano. Caso sejam necessárias mais habilidades para concluir a execução do plano, volta ao searchResource, caso já tenha executado todas as

habilidades, executa o GoToExit. Este pede o transporte para à saída e de seguida executa o takeDown de modo a remover o PA do DF.

5.1.4 *Resource Agent*

A classe RA, apresentada na figura 5.5, também *extends* a classe Agent do JADE.

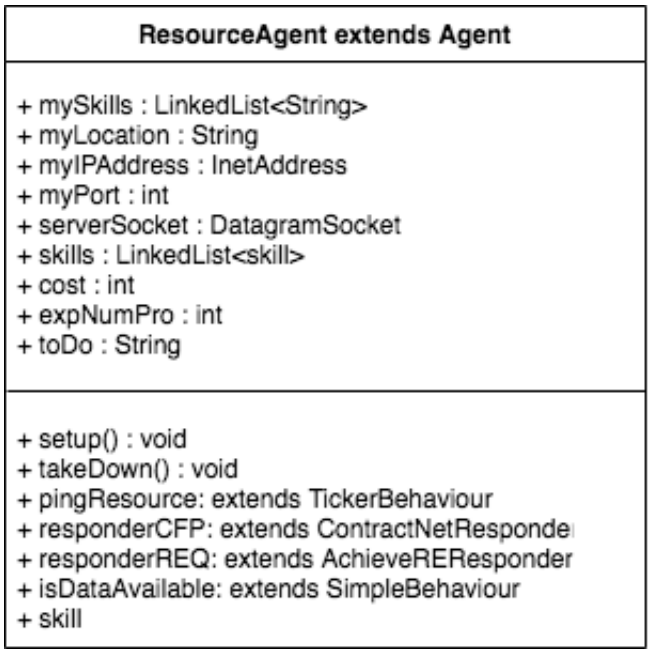


Figura 5.5: Classe *ResourceAgent*

Tal como no PA, a função setup inscreve o RA no DF com as suas habilidades e localização. O pingResource é um comportamento periódico que é responsável por verificar se o dispositivo de controlo do recurso ainda se encontra ligado. Caso o dispositivo tenha sido desligado, o RA executa a função takeDown, que remove o RA do DF.

O responderCFP é responsável por realizar a negociação de uma habilidade com o PA. Este ao receber um pedido de proposta envia o seu custo. Nesta implementação, o custo é um valor aleatório, pois o funcionamento do SMA não é o foco desta dissertação. Caso a proposta seja aceite, o RA envia a sua localização ao PA.

No responderREQ é tratada a receção de Requests do PA. Caso o PA peça para o RA enviar o custo, este envia uma mensagem com o seu custo. Caso peça para o RA executar uma habilidade, este comunica com o dispositivo de controlo para executar essa habilidade. Ao terminar, informa o PA do resultado.

O isDataAvailable é utilizado para receber a confirmação de que uma habilidade foi executada pelo dispositivo de controlo. De modo a não bloquear este agente, este comportamento corre até que tenha alguma informação a receber do dispositivo, altura em que lê essa informação, confirmando o final da execução de uma habilidade.

5.1.5 *Deployment Agent*

Tal como nas classes anteriores, o DA (fig. 5.6) também *extends* a classe Agent do JADE.

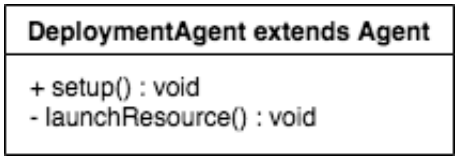


Figura 5.6: Classe *DeploymentAgent*

Esta classe contém também a função setup. Esta apenas chama a função launchResource, responsável por lançar recursos. A função espera que um módulo de produção se tente ligar ao sistema. Quando isto acontece, recebe a identificação, as habilidades e localização do recurso e tenta lançar o mesmo no SMA.

5.1.6 Comunicação entre Agentes

Na implementação dos agentes vista anteriormente é necessário considerar a comunicação entre agentes. As comunicações ocorrem entre PA e RA e PA e TA. A implementação destas comunicações é explicada de seguida. Para implementar as comunicações foram utilizados os protocolos FIPA Request e FIPA ContractNet [38], já explicados em 3.5.

5.1.6.1 Negociação entre *Product Agent* e *Resource Agent*

A negociação entre PA e RA, apresentada na figura 5.7, utiliza o protocolo FIPA Contract-Net. O PA, depois de ver no seu plano a habilidade que necessita de executar, envia um CFP (*Call For Proposals*) aos recursos com essa habilidade (skill) como parâmetro. Os RA podem recusar ou enviar uma proposta com o seu custo (cost). O PA compara todas as propostas e escolhe aquela com o custo mais baixo, aceitando essa e rejeitando as restantes. O RA escolhido, ao ver que o PA aceitou a proposta, pode recusar, ou aceitar e enviar a sua posição.

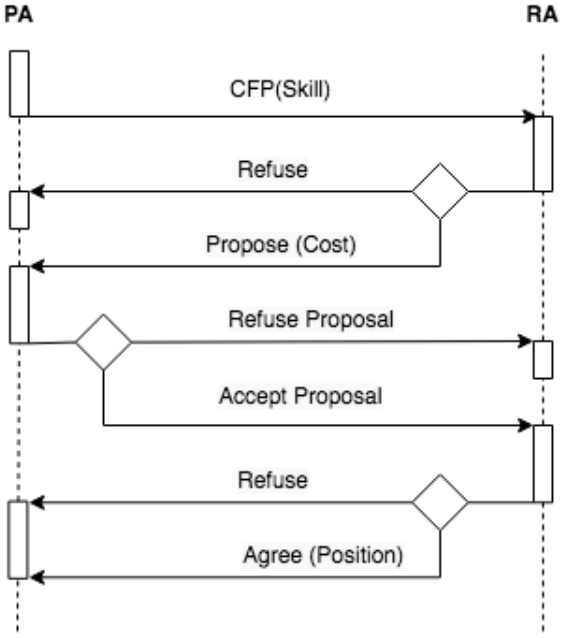


Figura 5.7: Diagrama de sequência da negociação entre PA e RA.

5.1.6.2 Pedido do *Product Agent* ao *Transport Agent*

Depois de negociar com o RA, o PA recebe a posição do RA escolhido e tem de recorrer ao TA para que o transporte. Para fazer este pedido, utiliza o protocolo FIPA Request. Assim, o PA envia um Req(*Request*) ao TA com a posição (position) recebida do RA como parâmetro. O TA pode recusar ou aceitar. Caso aceite, executa o transporte e informa o PA sobre o resultado. O resultado pode ser a informação de que algo falhou ou apenas uma informação (Inform) de que o transporte foi efetuado. Esta comunicação é apresentada na figura 5.8.

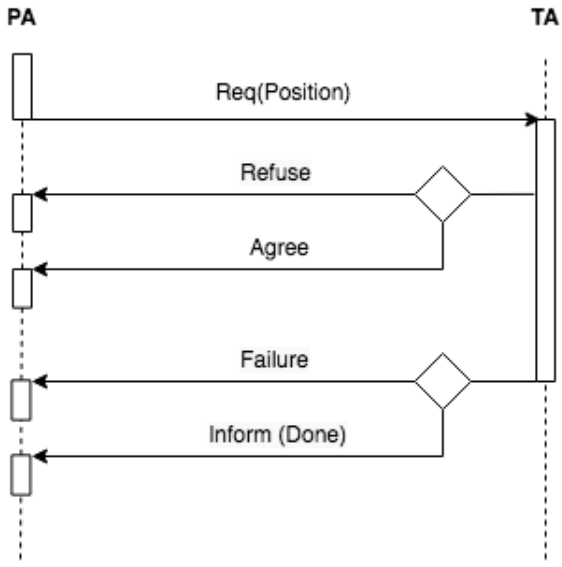


Figura 5.8: Diagrama de sequência do pedido de transporte do PA ao TA.

5.1.6.3 Pedido do *Product Agent* ao *Resource Agent*

Quando o PA chega à posição em que se encontra o RA escolhido para executar a habilidade pretendida, tem que fazer o pedido ao RA para que execute essa habilidade. Este pedido é feito de forma semelhante àquele mostrado anteriormente. É novamente utilizado o FIPA Request, mas no Req, em vez de enviar a posição, envia a habilidade (skill) pretendida. O RA pode recusar ou aceitar. No caso de aceitar informa o PA sobre o resultado da execução. A mesma pode falhar ou ter sucesso. Esta comunicação é apresentada na figura 5.9.

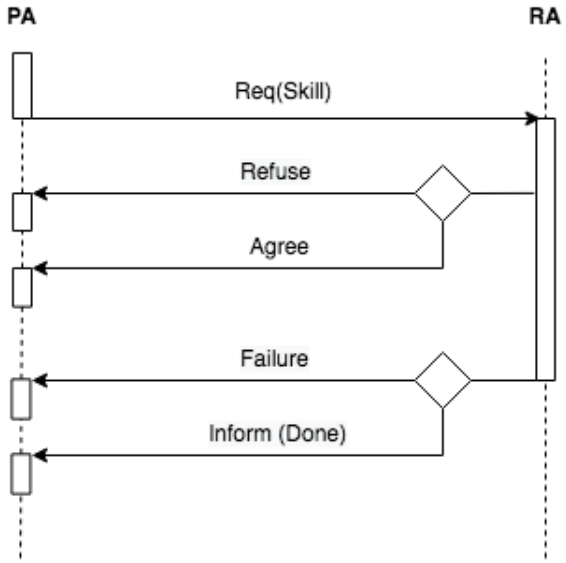


Figura 5.9: Diagrama de sequência do pedido de execução de uma habilidade do PA ao RA.

5.2 Dispositivo IoT

Tal como referido em 4.3, o dispositivo de controlo terá de ser capaz de se ligar através de rede *Ethernet* com a máquina que executa o SMA e de se ligar a um ou mais recursos físicos. A capacidade de processamento não tem de ser muito alta, pois não executa nenhum agente, mas tem de ser suficiente para controlar a execução de habilidades e para comunicar com SMA, permitindo a sua adição e remoção do sistema.

Existem algumas hipóteses de dispositivos no mercado que oferecem estas características e qualquer um poderia ser usado para implementar a arquitetura proposta. No entanto, esta implementação visa a utilização de dispositivos baseados em Arduino. Esta escolha deve-se à versatilidade e diversidade destes dispositivos.

A Arduino disponibiliza uma grande variedade de microcontroladores, assim como outro tipo de *hardware* que pode ser combinado com os microcontroladores, de modo a

criar soluções IoT mais completas. Existem ainda outras empresas que fabricam controladores baseados em Arduino, havendo assim uma grande variedade deste tipo de produtos no mercado.

Uma característica importante destes dispositivos é que tanto o seu *hardware*, como o *software* são *open-source*, possibilitando que qualquer pessoa possa desenvolver os seus projetos utilizando estes dispositivos.

Os microcontroladores são programáveis e a Arduino disponibiliza um ambiente de desenvolvimento próprio onde pode ser desenvolvido o *software* a ser executado nos dispositivos. Estes controladores têm sido cada vez mais utilizados na manufatura, sendo por isso interessante a sua utilização no desenvolvimento deste projeto.

5.3 Comunicação entre sistema multiagente e dispositivo

A comunicação entre SMA e dispositivo de controlo é feita em três momentos diferentes: quando um recurso se liga ao SMA, quando é necessário executar uma habilidade e ainda a verificação periódica se um recurso foi removido do sistema. Em todos foi utilizada a comunicação por sockets. Estas comunicações são explicadas de seguida.

5.3.1 Ligação do dispositivo ao sistema multiagente

Para o pedido de ligação do dispositivo de controlo ao SMA foram utilizados *datagram sockets*. Nesta comunicação, o dispositivo envia um pacote com o ID do recurso, as habilidades que este oferece e a sua localização no *shop-floor*.

A comunicação tem dois intervenientes, cliente (dispositivo de controlo) e servidor (DA), sendo que vários clientes podem comunicar com o mesmo servidor.

O cliente cria um pacote com a informação a enviar, o tamanho da informação e o IP e porta do servidor e faz o envio. De seguida fica à espera de receber um pacote do servidor com a confirmação de que foi lançado.

Por sua vez, o servidor cria um *datagram socket* numa determinada porta livre e fica à espera de receber informação de um cliente nessa porta. Ao receber a informação consegue identificar também o IP e porta do cliente e envia a confirmação de que lançou o recurso no sistema. De seguida, fica novamente à espera de receber informação.

Na figura 5.10 é apresentado o diagrama a exemplificar a integração de um dispositivo.

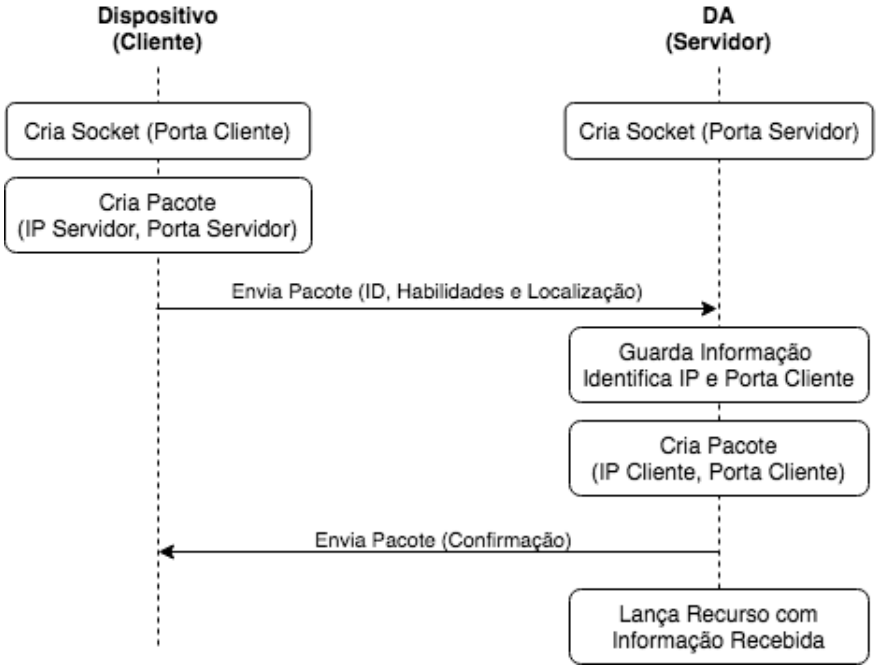


Figura 5.10: Diagrama de sequência do pedido de ligação de um dispositivo ao DA.

5.3.2 Execução de uma habilidade

No caso da execução de uma habilidade, foram utilizados *stream sockets*. Também nesta comunicação existe um servidor e um cliente. No entanto, neste caso, o cliente é o SMA e o servidor é o dispositivo de controlo.

De modo a pedir a execução de uma habilidade, o SMA (cliente) estabelece uma ligação com o IP e porta do dispositivo, as quais recebeu quando foi lançado pelo DA. Depois de ambos concordarem, podem trocar informação. Assim, o RA envia a informação ao dispositivo com a habilidade que pretende que o recurso físico execute. De seguida, fica à espera de receber a confirmação da execução.

O dispositivo de controlo (servidor), cria um *socket* numa determinada porta e fica à espera de receber um pedido de ligação. Ao receber o pedido com a habilidade a executar, este, ligado fisicamente ao recurso físico, atua sobre o mesmo, de modo a executar a habilidade. Ao acabar envia a confirmação ao RA e fica novamente à espera de receber um pedido de ligação.

Um exemplo desta comunicação é mostrado na figura 5.11.

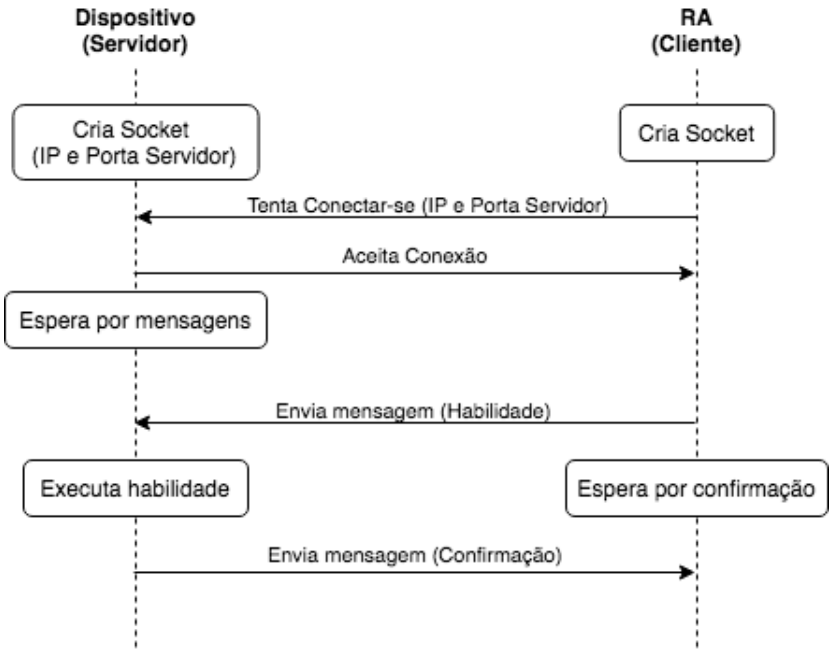


Figura 5.11: Diagrama de sequência do pedido de execução de uma habilidade do RA a um dispositivo.

5.3.3 Remoção de um dispositivo do sistema multiagente

De modo a manter o sistema de produção atualizado, é necessário verificar quando um recurso é removido fisicamente, tendo este de ser removido do DF para que não seja escolhido por um produto para executar uma habilidade, quando na verdade já não está disponível.

Esta verificação é feita de forma periódica pelo RA. O processo é muito parecido com o de execução de uma habilidade explicado anteriormente. Também aqui são utilizados *stream sockets*, sendo que é adicionado um tempo limite para estabelecer a ligação. Caso o RA não consiga estabelecer a ligação com o dispositivo de controlo nesse intervalo de tempo, o recurso é removido do DF. Caso a ligação seja bem sucedida é enviada uma mensagem de controlo, que é de seguida devolvida pelo dispositivo.

Esta verificação é feita com uma periodicidade de 3 segundos, sendo o tempo limite para o sucesso da ligação de 2 segundos. Um exemplo desta verificação é mostrado na figura 5.12.

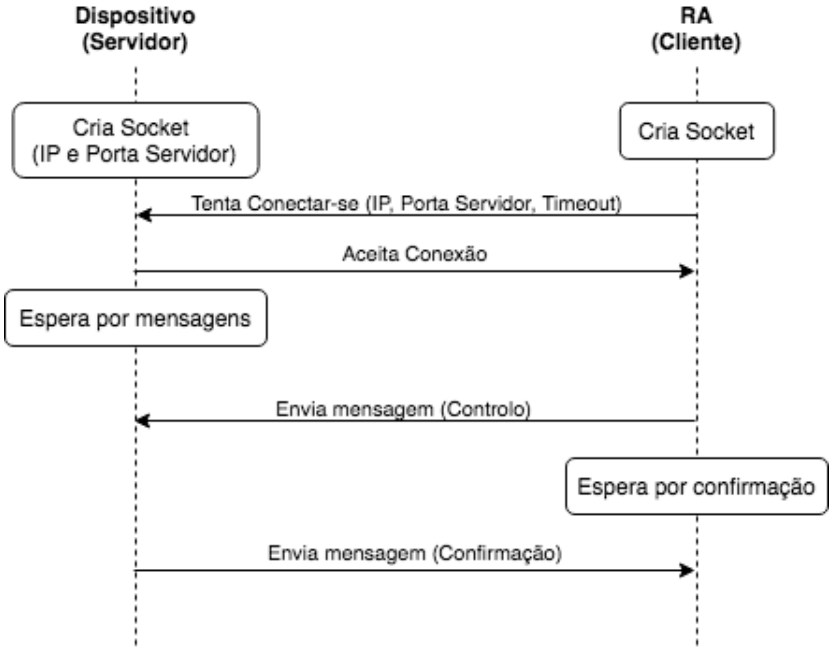


Figura 5.12: Diagrama de sequência do controlo do RA a um dispositivo.

Como se pode ver, estas comunicações são muito importantes para atingir o objetivo pretendido. Utilizando a comunicação por sockets torna-se possível a adição e remoção de módulos do sistema, bem como a interação necessária para a execução de uma habilidade.

Assim, o controlo do sistema é dividido entre SMA e dispositivo IoT. O SMA, através das interações entre os seus agentes, gere a adição e remoção de módulos do sistema, ao mesmo tempo que gere o plano de execução de cada produto, decidindo que recursos deve utilizar. O dispositivo, neste caso baseado em Arduino, apenas terá que controlar a execução das habilidades que lhe são requisitadas a cada momento pelo SMA.

Capítulo 6

Validação e Testes

De modo a validar a arquitetura e implementação propostas, foram executados três cenários de teste. No primeiro foi testada a velocidade de conexão de um dispositivo de controlo ao SMA e nos seguintes o desempenho de todo o sistema. Para estes testes foi utilizado um computador pessoal Apple MacBook Pro 2011, com processador 2,4 GHz Intel Core i5, memória RAM de 4 GB 1333 MHz DDR3 e 1 TB de disco para executar o SMA. Foram também utilizados três dispositivos de controlo (dois Arduinos e um PLC baseado em Arduino) e um kit de demonstração Staudinger GMBH como recurso físico. Os testes foram realizados em duas configurações de linha diferentes.

6.1 Configuração do Ambiente de Teste

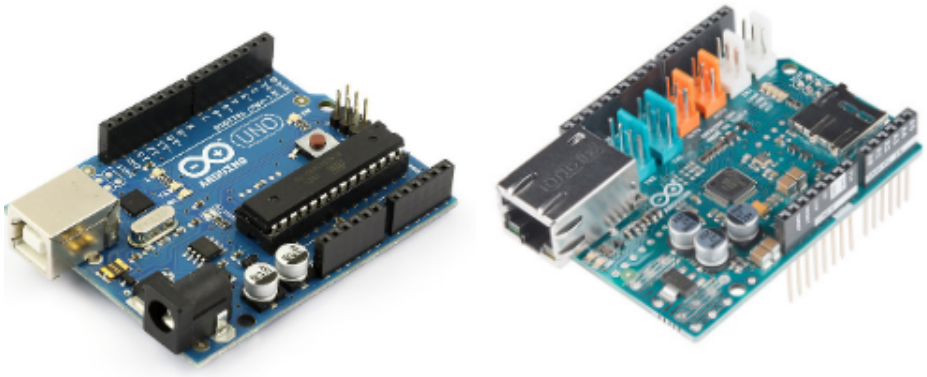
Tal como descrito anteriormente, o SMA foi alojado no computador pessoal, o qual foi ligado em rede com os dispositivos de controlo através de Ethernet. Por sua vez, os dispositivos IoT utilizados foram dois Arduino Uno [44] e um M-DUINO PLC ARDUINO ETHERNET 58 I/O's ANALOG/DIGITAL PLUS da Industrial Shields [45].

As especificações do Arduino Uno são apresentadas na tabela 6.1.

Tabela 6.1: Especificações do Arduino Uno [44]

	Arduino Uno
Microcontrolador	ATmega328P
Tensão de operação	5 V
Pinos digitais I/O	14
Pinos digitais PWM I/O	6
Pinos analógicos de entrada	6

Uma vez que o Arduino Uno não suporta comunicação Ethernet e esta é necessária para se ligar à rede local, foi necessário adicionar uma Arduino Ethernet Shield 2 capaz de fazer essa ligação. Estes dois componentes são apresentados na figura 6.1.



(a) Arduino Uno. (b) Arduino Ethernet Shield 2.

Figura 6.1: Conjunto Arduino Uno e Arduino Ethernet Shield 2.

Como o kit utilizado para à demonstração opera a uma tensão de 24V e o Arduino Uno opera a uma tensão de 5V, seria necessário adicionar mais *hardware* para se poder controlar o kit. Assim, para não aumentar a complexidade da ligação entre dispositivo de controlo e recurso físico e porque esta ligação não é o foco desta dissertação, foi decidido que no Arduino Uno, as habilidades oferecidas seriam simuladas utilizando a função *delay()*, a qual suspende a execução do Arduino durante um período de tempo.

Para controlar o kit de demostração utilizou-se então o M-DUINO PLC ARDUINO ETHERNET 58 I/O's ANALOG/DIGITAL PLUS da Industrial Shields, cujas especificações são apresentadas na tabela 6.2. Este PLC é baseado no Arduino Mega 2560, sendo que ambos têm o mesmo controlador, oferecendo o mesmo desempenho.

Tabela 6.2: Especificações do M-DUINO PLC ARDUINO ETHERNET 58 I/O's ANALOG/-DIGITAL PLUS [45].

	M-DUINO ETHERNET 58 PLUS
Microcontrolador	ATmega2560
Tensão de operação	Entre 11.4V e 25.4V
Pinos de Entrada	36
Pinos de Saída	22

Este PLC está dividido em 4 zonas, como se vê na figura 6.2. A zona A é uma *shield* de comunicação, a qual oferece uma entrada Ethernet utilizada para ligar o PLC à rede local, assim como uma entrada USB, utilizada apenas para programar o PLC. As restantes zonas são as entradas/saídas analógicas e digitais.

Figura 6.2: ETHERNET PLC M-DUINO 58+ e distribuição de zonas.

Como recurso físico foi utilizado um kit da Staudinger GMBH, o qual é apresentado na figura 6.3. Este kit é constituído por 4 passadeiras (P1,P2,P3 e P4, da esquerda para a direita) e dois módulos de produção, M1 (esquerda) e M2 (direita).

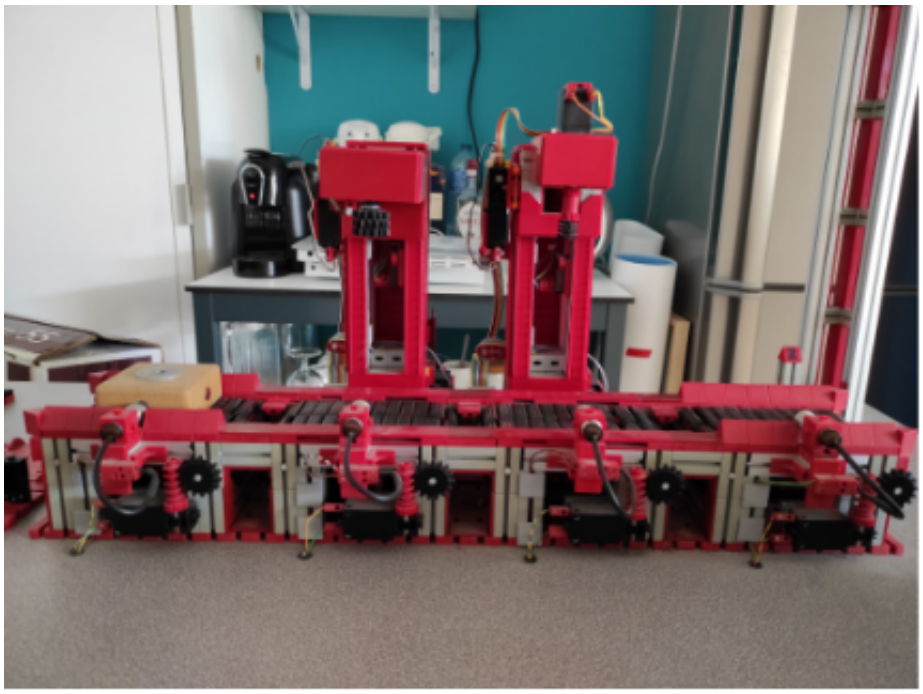
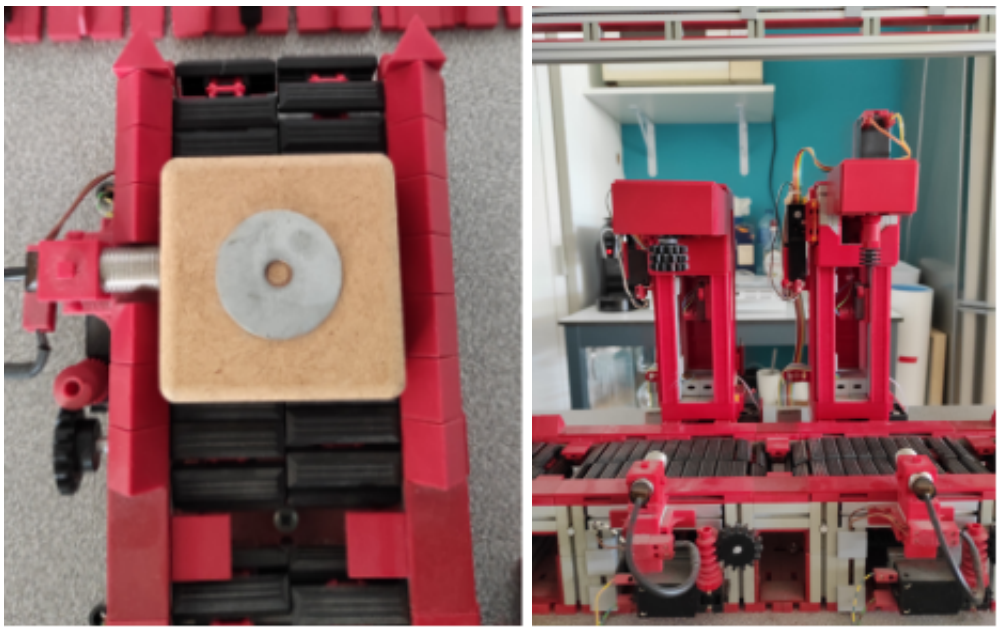


Figura 6.3: Kit de Demonstração da Staudinger GMBH.

Cada passadeira (figura 6.4(a)) tem um sensor que deteta quando uma peça está na passadeira e um motor de duas direções.

Os módulos (figura 6.4(b)) são semelhantes, sendo que ambos se movem horizontalmente e verticalmente, tendo depois uma ferramenta diferente para atuar sobre o produto. Possuem ainda sensores mecânicos utilizados para verificar os limites de uma deslocação e os quais podem ser movidos para definir a amplitude dos movimentos do módulo.



(a) Passadeira. (b) Módulos.

Figura 6.4: Passadeira e Módulos de Produção

6.2 Teste de Velocidade de Conexão

O primeiro teste a ser realizado foi o teste de velocidade de conexão de um dispositivo de controlo ao SMA. Como visto anteriormente, quando um dispositivo é ligado à alimentação, este envia imediatamente um pedido ao DA, enviando a sua localização e habilidades, para que seja lançado no sistema como um RA e assim fique disponível para ser consumido pelo PA.

Para este teste foi utilizado o PLC referido na secção anterior. Com computador e PLC ligados na rede local, o SMA foi iniciado no computador e de seguida foi ligado o PLC à alimentação. Foi lido o instante em que o DA recebeu o pedido de ligação do PLC e posteriormente foi lido o instante em que o RA associado ao PLC foi registado na DF, ficando assim disponível para ser utilizado.

Registou-se a diferença entre os dois instantes, o que representou uma amostra. De seguida o PLC foi desligado da alimentação, tendo o RA associado sido removido automaticamente do sistema. Após ser removido, voltou a ligar-se o PLC, tendo este processo sido repetido até obter 40 amostras.

Destas amostras foram retiradas as três maiores e as três menores e de seguida foi calculada a média, a variância e o desvio padrão. A tabela 6.3 apresenta os resultados deste teste.

Tabela 6.3: Resultados do teste de velocidade de conexão.

Número de Amostras	Mínimo (ms)	Máximo (ms)	Média (ms)	Variância (ms)	Desvio Padrão (ms)
34	7	12	8,765	1,474	1,214

Pelos resultados apresentados vê-se que o tempo que decorre desde que um dispositivo de controlo deste tipo é detetado, até que esteja disponível para ser utilizado, é bastante curto. Assim, é possível concluir que a utilização destes dispositivos num sistema deste tipo é válida, pois o tempo de ligação permite que a adição e remoção de recursos seja feita de forma rápida, tornando o sistema escalável.

6.3 Testes de Desempenho do Sistema

De seguida foram executados dois cenários de teste ao funcionamento do sistema. Em ambos os cenários, o SMA foi alocado no computador, o qual foi ligado em rede com os dispositivos de controlo.

Foi utilizado um ambiente simulado que oferece duas configurações de linha diferentes. Este ambiente simula o sistema de transporte de cada linha bem como as estações virtuais que cada uma das linhas oferece, tornando assim possível ver o trajeto de um produto ao longa de cada uma das linhas. Quando um produto chega a uma estação, o SMA faz o pedido ao dispositivo de controlo, o qual interage com o kit de demonstração, para que execute a habilidade acordada, podendo essa execução ser vista em ambiente real. Após a habilidade ser executada pode ver-se novamente o trajeto do produto através da linha no ambiente simulado.

6.3.1 Teste com Configuração de Linha 1

Para o primeiro teste foi utilizada a configuração de linha apresentada na figura 6.5. Nesta configuração, um produto apenas pode ser transportado para uma das estações disponíveis na linha, sendo de seguida encaminhado para a saída.

Para este teste foram utilizados dois dispositivos de controlo, o PLC e o Arduino Uno. Ao PLC foi também ligado o kit de demonstração. Assim, foi definido que o Arduino seria o recurso 1 (R1) e que iria executar a habilidade atómica A (H_A) e que o PLC seria o recurso 2 (R2) e que iria executar também a H_A.

Como no Arduino foi decidido que a execução de habilidades seria simulada com a função *delay*, foi medido o tempo de execução da H_A no kit de demonstração e atribuído o mesmo tempo ao *delay* para que os resultados obtidos fossem concordantes. Neste caso a H_A corresponde a um *delay* de 16 segundos.

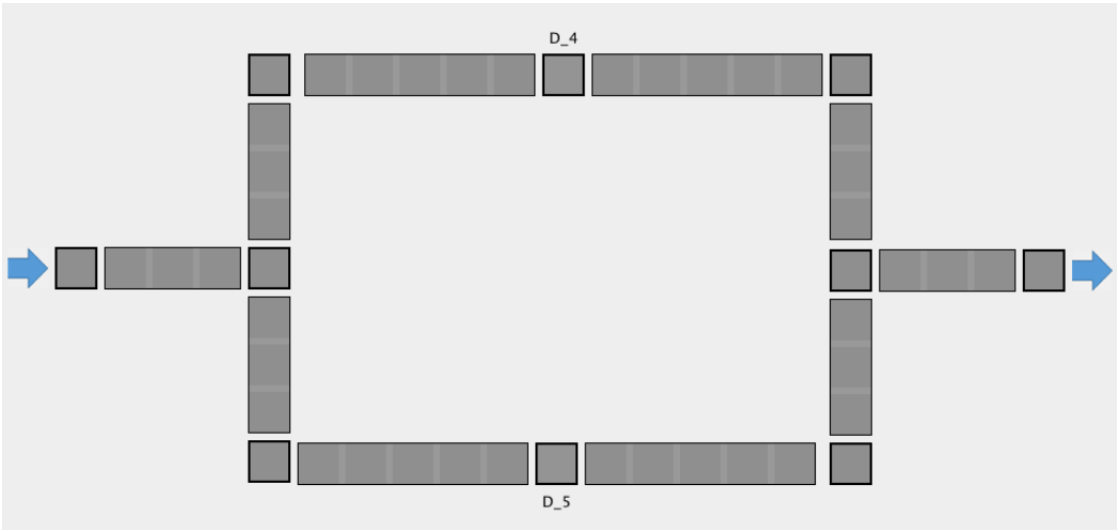


Figura 6.5: Configuração de linha 1.

Na figura 6.6 pode ser visto a sequência de imagens correspondente à execução da H_A no kit.

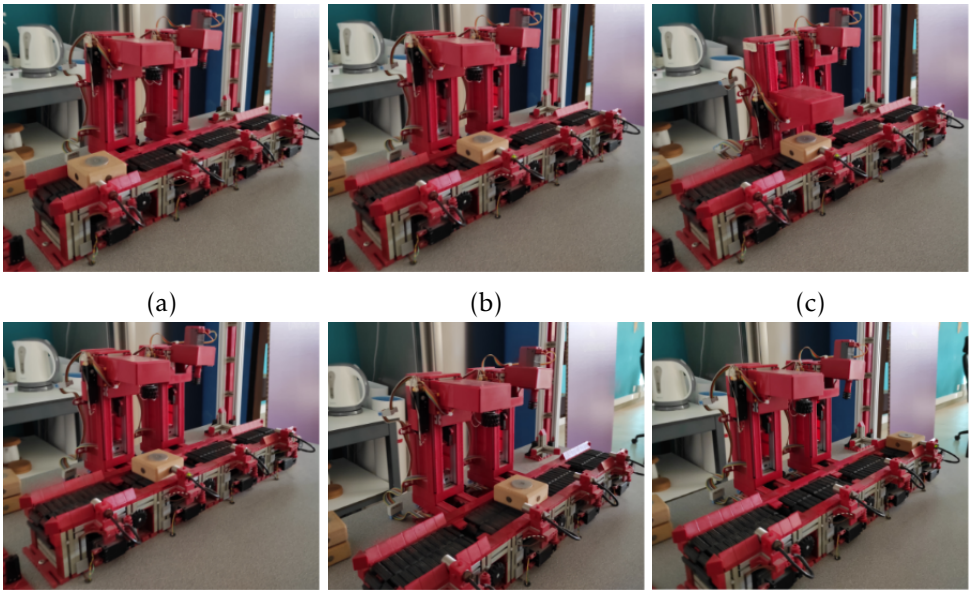


Figura 6.6: Execução da H_A no kit.

Depois de colocado um produto no centro da passadeira P1, o kit segue o plano de execução representado no fluxograma da figura 6.7.

Figura 6.7: Fluxograma do plano de execução de H_A no kit de demonstração.

Este cenário de teste consiste em iniciar o SMA no computador e de seguida ligar o Arduino à alimentação. Como já explicado em capítulos anteriores, o Arduino pede ao SMA para se ligar, enviando a sua localização e habilidades que oferece. Neste caso, a localização é D_5 (corresponde à posição D_5 na configuração de linha 1) e a habilidade é H_A. Uma vez lançado no sistema, o R1 fica disponível para ser consumido por produtos.

De seguida são lançados 5 produtos ao mesmo tempo no computador, os quais têm como plano de execução a habilidade H_A. Neste caso, como apenas existe um recurso no sistema que oferece esta habilidade, é esperado que todos os 5 produtos sejam encaminhados para D_5, que seja executada a habilidade nessa estação e que de seguida sejam encaminhados para a saída.

Depois de produzidos estes 5 produtos, é ligado o PLC à alimentação. Tal como o Arduino, este pede ao SMA para se ligar, enviando a localização, D_4, e a habilidade que oferece, H_A. Depois de lançado o R2 no sistema, passam a existir dois recursos em localizações diferentes que oferecem a mesma habilidade.

São então lançados novamente 5 produtos tendo como plano de execução a H_A, sendo que desta vez, os mesmos serão divididos entre D_4 e D_5, otimizando assim o seu tempo de produção. Após este passo, é desligado o Arduino, sendo o R1 eliminado do sistema. São novamente lançados 5 produtos, com o mesmo plano de execução e verifica-se que os 5 são encaminhados para D_4.

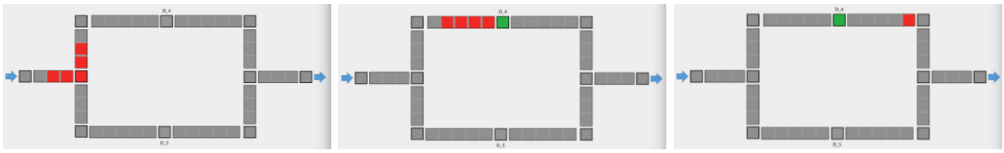
Nas figuras 6.8, 6.9 e 6.10 é apresentada a evolução destes testes como visto no simulador.

(a) (b) (c)

Figura 6.8: Evolução do primeiro conjunto de 5 produtos no simulador.

(a) (b) (c)

Figura 6.9: Evolução do segundo conjunto de 5 produtos no simulador.



(a) (b) (c)

Figura 6.10: Evolução do terceiro conjunto de 5 produtos no simulador.

Durante este teste foram medidos os tempos de produção de cada um destes produtos para uma melhor análise dos resultados. Assim, nas tabelas em baixo são apresentados os resultados do teste. Os resultados de cada um dos conjuntos de 5 produtos foram divididos em diferentes tabelas para uma melhor comparação e análise.

Tabela 6.4: Resultados do primeiro conjunto de 5 produtos.

Produtos	**Recurso Utilizado**	**Tempo de Produção**
P1	R1	45
P2	R1	61
P3	R1	78
P4	R1	94
P5	R1	111

Tabela 6.5: Resultados do segundo conjunto de 5 produtos.

Produtos	**Recurso Utilizado**	**Tempo de Produção**
P6	R2	44
P7	R1	44
P8	R2	60
P9	R1	60
P10	R2	76

Tabela 6.6: Resultados do terceiro conjunto de 5 produtos.

Produtos	**Recurso Utilizado**	**Tempo de Produção**
P11	R2	45
P12	R2	61
P13	R2	78
P14	R2	94
P15	R2	111

Assim, observando as tabelas e as imagens, é possível verificar que os resultados foram os esperados.

Na primeira tabela (6.4) existe uma diferença de 16 segundos entre os tempos de produção dos produtos, o que corresponde ao tempo de execução da H_A. Todos os produtos utilizaram o mesmo recurso, como apresentado na figura 6.8.

Na segunda tabela (6.5), continua a existir uma diferença de 16 segundos entre os tempos de produção, no entanto, esta diferença apenas se reflete de dois em dois produtos, uma vez que o sistema passou a ter a capacidade de produzir dois produtos de uma vez, com a adição do R2. Na figura 6.9 pode ser vista a distribuição dos produtos pelos dois recursos existentes.

Por último, na terceira tabela (6.6), verificam-se novamente os tempos da primeira tabela, o que era esperado, uma vez que houve a remoção do R1, ficando novamente só um recurso disponível para ser utilizado por todos os produtos, como é confirmado pela figura 6.10.

6.3.2 Teste com Configuração de Linha 2

Neste teste foi utilizada uma configuração de linha mais complexa, baseada na linha da NovaFlex, a qual permite simular uma linha de produção de um sistema industrial e que é apresentada na figura 6.11. Com esta configuração os produtos podem viajar entre diferentes estações até que todas as habilidades do seu plano tenham sido executadas.

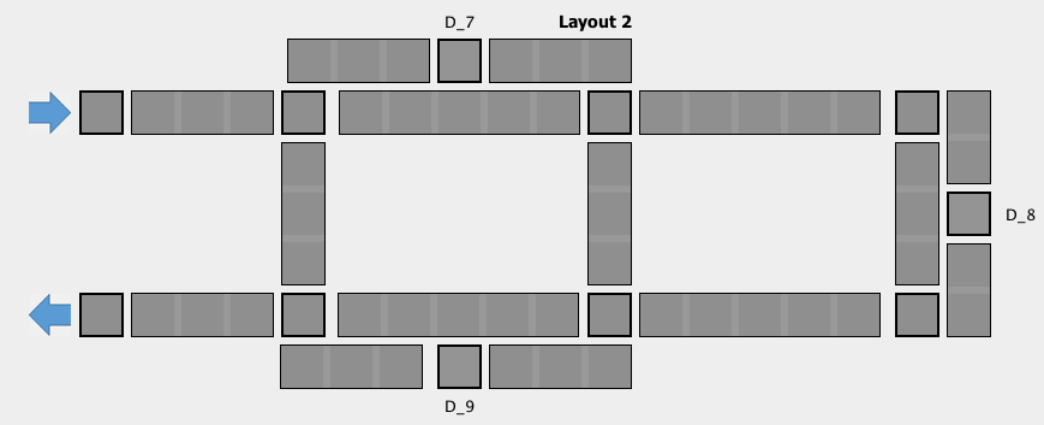


Figura 6.11: Configuração de linha 2.

Foram utilizados os três dispositivos de controlo, dois Arduino Uno e o PLC.

Neste caso, o Arduino Uno utilizado no teste anterior corresponde ao recurso 1 (R1) e executa a habilidade B (H_B) para além da H_A. Esta habilidade consiste num *delay* de 5 segundos. O outro Arduino, que corresponde ao recurso 2 (R2), executa a habilidade C (H_C), a qual consiste num *delay* de 16 segundos.

Finalmente, o PLC é considerado o recurso 3 (R3) e executa a H_C. No caso do PLC, esta habilidade representa uma ação no kit, cujo plano é apresentado na figura 6.12.

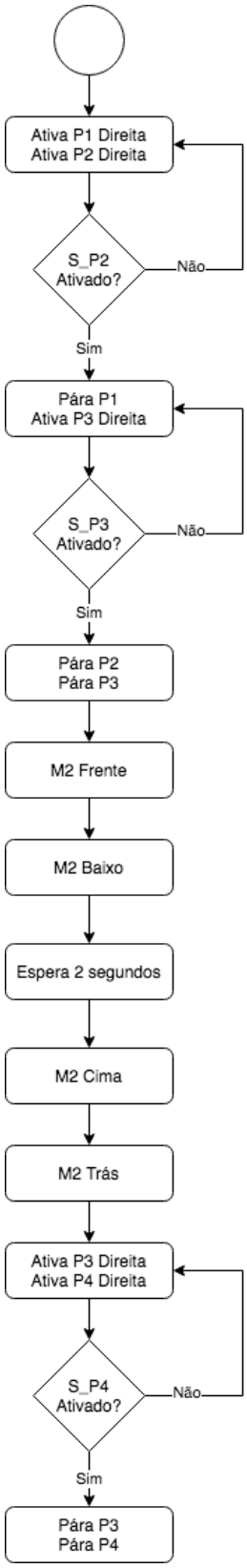


Figura 6.12: Fluxograma do plano de execução de H_C no kit de demonstração.

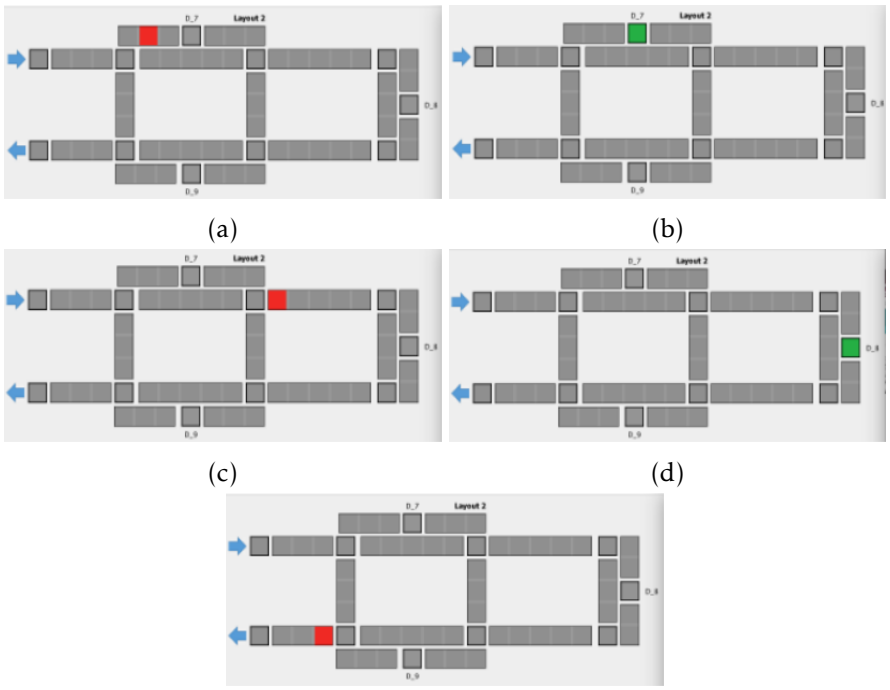
Assim, mais uma vez, é iniciado o SMA no computador e de seguida, ambos os Arduinos são ligados à alimentação. O R1 pede para se ligar, enviando as habilidades disponíveis (H_A e H_B) e localização (D_7), assim como o R2, com a H_C e localização D_8.

De seguida é lançado um produto, cujo plano de produção corresponde à execução de H_B e H_C. Este produto deve entrar na linha, seguir para a estação D_7, onde é executada a H_B, de seguida deve seguir para a estação D_8, onde é executada a H_C, sendo depois encaminhado para a saída.

Depois de executado o plano de produção do primeiro produto, observa-se que a estação D_9 fica mais próxima de D_7 que a estação D_8. Assim, é necessária uma reconfiguração da linha. É então desligado o R2 e ligado o R3, o qual ao ligar-se, oferece as habilidades H_A e H_C na localização D_9.

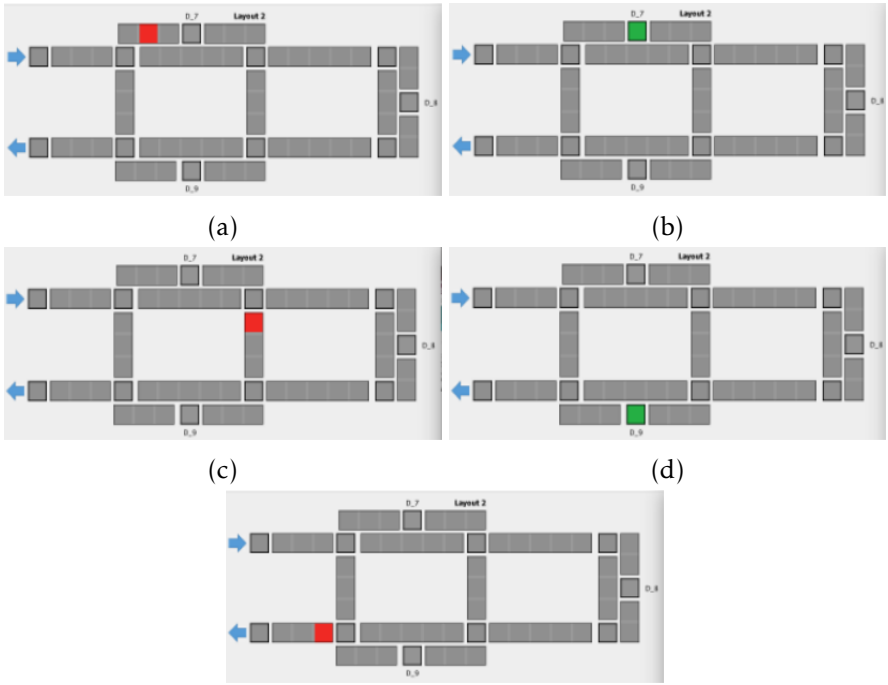
A seguir é lançado novamente um produto com o mesmo plano de produção que o anterior, o qual deverá viajar para D_7, onde é executada a H_B, e de seguida, deverá deslocar-se para D_9 para que seja executada a H_C, sendo depois encaminhado para a saída.

Nas figuras 6.13 e 6.14 é apresentada a realização do teste como visto no simulador, apresentando os vários passos da execução.



(a) (b) (c) (d) (e)

Figura 6.13: Evolução do primeiro produto no simulador.



(a) (b) (c) (d) (e)

Figura 6.14: Evolução do segundo produto no simulador.

Pelas figuras 6.13 e 6.14 é possível verificar que os resultados foram os esperados, ficando assim provada a reconfigurabilidade do sistema.

Em suma, os testes foram bem sucedidos. No primeiro teste os resultados mostram que é possível uma rápida adição de um dispositivo ao sistema, tendo os valores sido constantes. Nos dois cenários seguintes foi possível validar a arquitetura e implementação propostas, tendo sido possível a adição e remoção de módulos, assim como a execução de habilidades.

Capítulo 7

Conclusões e Trabalho Futuro

7.1 Conclusões

Esta dissertação foca-se nos desafios impostos pelas novas exigências dos mercados impostos ao setor da manufatura. Foram analisados os novos paradigmas de produção, os quais apareceram na tentativa de responder a estas exigências, tendo tido no entanto algumas dificuldades.

Assim, foi proposta uma arquitetura que divide o controlo entre um sistema multiagente e um dispositivo IoT, tentando assim ultrapassar as barreiras que o *hardware* coloca na tentativa de implementação destes paradigmas.

Esta arquitetura foi implemetada com a utilização de dispositivos Arduino, sendo a comunicação com o SMA feita com sockets. A implementação desta comunicação foi o maior desafio encontrado durante a realização deste trabalho, sendo também a característica chave ao funcionamento da arquitetura.

No final foi testada a validade desta solução recorrendo a dois microcontroladores diferentes e a um kit de demonstração, sendo possível concluir pelos resultados obtidos que a utilização de microcontroladores integradores com um sistema multiagente na tentativa de obter um sistema de produção flexível é uma solução válida e que poderá ser explorada.

Utlizando a arquitetura e implementação apresentadas foi possível responder à pergunta apresentada inicialmente, tendo sido apresentada uma arquitetura que possibilita a integração de módulos de produção simples de baixo custo, não envolvidos no processo de decisão, de forma rápida e eficiente, num sistema de controlo distribuído e adaptável.

Os resultados obtidos indicam que é possível atingir um sistema com capacidade para responder às exigências atuais dos mercados, permitindo a produção de produtos personalizáveis de modo eficiente e a um baixo custo.

O teste de velocidade de conexão mostra que os resultados são promissores, sendo no entanto necessários testes adicionais num ambiente de produção real.

Pelos restantes testes pode também concluir-se que a utilização de microcontroladores Arduino como facilitadores da característica *Plug&Produce* é possível. Ao mesmo tempo, esta dissertação não limita a arquitetura à utilização de dispositivos deste tipo, ficando as linhas gerais para à implementação com outro tipo de dispositivos IoT na tentativa de obter um sistema escalável e reconfigurável.

7.2 Trabalho Futuro

Pelos resultados obtidos foi possível demonstrar a validade desta solução. No entanto, existe ainda algum trabalho necessário para que esta solução possa ser utilizada na indústria.

Assim, seria interessante desenvolver testes adicionais recorrendo a um demonstrador real, primeiro utilizado em testes em laboratório, e mediante os resultados obtidos, seria testado em ambiente industrial.

Como referido, esta solução utilizou dispositivos baseados em Arduino na implementação da arquitetura. Seria no entanto interessante a validação desta arquitetura recorrendo a outro tipo de microcontroladores disponíveis no mercado e sendo feita posteriormente uma comparação dos resultados obtidos.

Bibliografia

[1] R. Y. Zhong, X. Xu, E. Klotz e S. T. Newman. "Intelligent manufacturing in the context of industry 4.0: a review". Em: *Engineering* 3.5 (2017), pp. 616–630.

[2] A. Rojko. "Industry 4.0 concept: background and overview". Em: *International Journal of Interactive Mobile Technologies (iJIM)* 11.5 (2017), pp. 77–90.

[3] Y. Lu. "Industry 4.0: A survey on technologies, applications and open research issues". Em: *Journal of industrial information integration* 6 (2017), pp. 1–10.

[4] L. Ribeiro. "Cyber-physical production systems' design challenges". Em: *2017 ieee 26th international symposium on industrial electronics (isie)*. IEEE. 2017, pp. 1189–1194.

[5] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn e K. Ueda. "Cyber-physical systems in manufacturing". Em: *Cirp Annals* 65.2 (2016), pp. 621–641.

[6] J. Lee, B. Bagheri e H.-A. Kao. "A cyber-physical systems architecture for industry 4.0-based manufacturing systems". Em: *Manufacturing letters* 3 (2015), pp. 18–23.

[7] M. P. Groover. *Automation, production systems, and computer-integrated manufacturing*. Pearson Education India, 2016.

[8] T. Raj, R. Shankar e M. Suhaib. "A review of some issues and identification of some barriers in the implementation of FMS". Em: *International Journal of Flexible Manufacturing Systems* 19.1 (2007), pp. 1–40.

[9] M. G. Mehrabi, A. G. Ulsoy, Y. Koren e P. Heytler. "Trends and perspectives in flexible and reconfigurable manufacturing systems". Em: *Journal of Intelligent manufacturing* 13.2 (2002), pp. 135–146.

[10] Y. Koren, U Heisel, F Jovane, T Moriwaki, G Pritschow, G Ulsoy e H Van Brussel. "Reconfigurable manufacturing systems". Em: *Annals of the CIRP* 48 (1999), p. 2.

[11] V. Marik e D. McFarlane. "Industrial adoption of agent-based technologies". Em: *IEEE Intelligent Systems* 20.1 (2005), pp. 27–35.

[12] L. Monostori, J. Váncza e S. R. Kumara. "Agent-based systems for manufacturing". Em: *CIRP annals* 55.2 (2006), pp. 697–720.

[13] F. Almeida, B. Terra, P. Dias e G. Goncalves. "Adoption issues of multi-agent systems in manufacturing industry". Em: *2010 Fifth International Multi-conference on Computing in the Global Information Technology*. IEEE. 2010, pp. 238–244.

[14] F. L. Bellifemine, G. Caire e D. Greenwood. *Developing multi-agent systems with JADE*. Vol. 7. John Wiley & Sons, 2007.

[15] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser e A. W. Colombo. "Smart agents in industrial cyber–physical systems". Em: *Proceedings of the IEEE* 104.5 (2016), pp. 1086–1101.

[16] S. Briickner, J. Wyns, P. Peeterst e M. J. Kollingbaum. "Designing Agents for Manufacturing Control". Em: 1998.

[17] L. Ribeiro, J. Barata, M. Onori e J. Hoos. "Industrial agents for the fast deployment of evolvable assembly systems". Em: *Industrial Agents*. Elsevier, 2015, pp. 301–322.

[18] PRIME. *Plug and PRoduce Intelligent Multi Agent Environment based on Standard Technology*. URL: https://cordis.europa.eu/project/id/314762.

[19] J. CHRISTENSEN. "Holonic manufacturing systems-initial architecture and standards directions". Em: *1st Euro. Conf. on HMS, Hannover, Germany, 1994*. 1994.

[20] R. F. Babiceanu e F. F. Chen. "Development and applications of holonic manufacturing systems: a survey". Em: *Journal of Intelligent Manufacturing* 17.1 (2006), pp. 111–131.

[21] P. Leitão e F. Restivo. "ADACOR: A holonic architecture for agile and adaptive manufacturing control". Em: *Computers in industry* 57.2 (2006), pp. 121–130.

[22] A Tharumarajah, A. Wells e L Nemes. "Comparison of emerging manufacturing concepts". Em: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*. Vol. 1. IEEE. 1998, pp. 325–331.

[23] M. Onori e J. Barata. "Evolvable production systems: mechatronic production equipment with process-based distributed control". Em: *IFAC Proceedings Volumes* 42.16 (2009), pp. 80–85.

[24] F. Jammes e H. Smit. "Service-oriented paradigms in industrial automation". Em: *IEEE Transactions on industrial informatics* 1.1 (2005), pp. 62–70.

[25] L. Ribeiro, J. Barata e P. Mendes. "MAS and SOA: complementary automation paradigms". Em: *International Conference on Information Technology for Balanced Automation Systems*. Springer. 2008, pp. 259–268.

[26] W. W. W. Consortium et al. "Web Services Architecture, W3C Working Group Note 11 February 2004". Em: *http://www. w3. org/TR/2004/NOTE-ws-arch-20040211/* (2004).

[27] F. Jammes, H. Smit, J. L. M. Lastra e I. M. Delamer. "Orchestration of service-oriented manufacturing processes". Em: *2005 IEEE conference on emerging technologies and factory automation*. Vol. 1. IEEE. 2005, 8–pp.

[28] H. Bohn, A. Bobek e F. Golatowski. "SIRENA-Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains". Em: *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*. IEEE. 2006, pp. 43–43.

[29] D Mourtzis, E Vlachou e N. Milas. "Industrial Big Data as a result of IoT adoption in manufacturing". Em: *Procedia cirp* 55 (2016), pp. 290–295.

[30] F. Chen, C. Ren, Q. Wang e B. Shao. "A process definition language for Internet of Things". Em: *Proceedings of 2012 IEEE International Conference on Service Operations and Logistics, and Informatics*. IEEE. 2012, pp. 107–110.

[31] J. Soldatos, S. Gusmeroli, P. Malo e G. Di Orio. "Internet of things applications in future manufacturing". Em: *Digitising industry-internet of things connecting the physical, digital and virtual worlds* (2016), pp. 153–183.

[32] C. Yang, W. Shen e X. Wang. "Applications of Internet of Things in manufacturing". Em: *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE. 2016, pp. 670–675.

[33] E. Sisinni, A. Saifullah, S. Han, U. Jennehag e M. Gidlund. "Industrial internet of things: Challenges, opportunities, and directions". Em: *IEEE Transactions on Industrial Informatics* 14.11 (2018), pp. 4724–4734.

[34] P. Mell e T. Grance. "Perspectives on cloud computing and standards". Em: *Usa, Nist* (2009).

[35] H. Ruschel, M. S. Zanotto e W. d. Mota. "Computação em nuvem". Em: *Pontifícia Universidade Católica do Paraná, Curitiba, Brazil* (2010).

[36] F. Tao, L. Zhang, V. Venkatesh, Y Luo e Y. Cheng. "Cloud manufacturing: a computing and service-oriented manufacturing model". Em: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 225.10 (2011), pp. 1969–1976.

[37] F. Bellifemine, F. Bergenti, G. Caire e A. Poggi. "JADE—a java agent development framework". Em: *Multi-agent programming*. Springer, 2005, pp. 125–147.

[38] FIPA. *The Foundation for Intelligent Physical Agents*. URL: http://www.fipa.org/.

[39] F. Bellifemine, A. Poggi e G. Rimassa. "Developing multi-agent systems with JADE". Em: *International Workshop on Agent Theories, Architectures, and Languages*. Springer. 2000, pp. 89–103.

[40] F. XC00026. "Fipa request interaction protocol specification". Em: *Foundation for Intelligent Physical Agents* (2002).

[41] F. XC00029. "Fipa contract net interaction protocol specification". Em: *Foundation for Intelligent Physical Agents* (2002).

[42] A. S. Tanenbaum e D. J. Wetherall. *Computer Networks*. 5th. USA: Prentice Hall Press, 2010. ISBN: 0132126958.

[43] J. S. Quarterman, A. Silberschatz e J. L. Peterson. "4.2 BSD and 4.3 BSD as examples of the UNIX system". Em: *ACM Computing Surveys (CSUR)* 17.4 (1985), pp. 379–418.

[44] Arduino. *ARDUINO UNO REV3*. URL: https://store.arduino.cc/arduino-uno-rev3.

[45] I. Shields. *M-DUINO PLC Arduino Ethernet 58 I/Os Analog/Digital PLUS*. URL: https://www.industrialshields.com/shop/product/is-mduino-58-m-duino-plc-arduino-ethernet-58-i-os-analog-digital-plus-176.